# Cryptology Annual News Update and Vignette

Bill Ricker
for BLU.org
Oct 16, 2024

1. Cryptology News Bulletins
2. Post Quantum Cryptograpy update
3. Historic Vignette
4. Bibliography

## §1 Cryptology News Bulletins 2023-09 to 2024-09

"**Abundance of Caution**" is C-suite lingo for "*Oopsie, oh flying squirrel*"

---

### *Fedora: OpenSSL now distrusts SHA-1*

- OpenSSL now distrusts SHA-1 for PKCS #1 signatures *by default* in Fedora41.
- this is a `openssl.cnf` option
  - which is set in Fedora41
  - probably advisable to set this on other systems?

### NOTES

- (this has been in works for 6+ years)

---

### *FROST Threshold Signature IRTF proposal*

June - RFC 9591 "FROST Flexible round-optimied Shnorr threshold signing protocol"

A cryptological protocol to allow a quorum of a distributed committee to sign a document.

- RFC 9591
- Informational RFC; not Standards Track (yet?)
- Evolution of "FROST: Flexible Round-Optimized Schnorr Threshold Signatures", December 2020
- requires prior out-of-band secure distribution of **n** key fragments, **m** of which will be needed for a valid signature.
- not PQC; relies on Discrete Log

---

## *End of End to End ?*

> Matthew Green @matthew_d_green 2024-05-02

"Europe is maybe two months from passing laws that end private communication as we know it, and folks are looking the other way (understandably.) You're not going to get a do-over once these laws are passed."

Xhitter

- Related
  - 2024-09-xx Australia passed such a law in 2018, and in Sept. 2024 threatened to start invoking it
  - 2023-09-xx Signal vowed to withdraw from UK if mandatory backdoors bill passed techcrunch
  - 2024-02-14 European Court of Human Rights rules mandatory backdoors in message services against EU Human Rights
    - https://www.schneier.com/blog/archives/2024/02/eu-court-of-human-rights-rejects-encryption-backdoors.html
    - Ars
    - EU Court
  - 2023-10-05 1994 CALEA mandated backdoor
    - WSJ {paywall} "U.S. Wiretap Systems Targeted in China-Linked Hack / AT&T and Verizon are among the broadband providers that were breached"
    - TechCrunch CNN WaPo
  - 2024-02-26 Nevada sues [Meta] to deny kids access to Messenger encryption
  - 2023-12-05 Discussion - Center for European Policy Analysis (CEPA) - "Encryption: It's Not About Good and Bad Guys, It's About All of Us" "Most digital messages are encrypted to protect privacy and security. Governments around the globe seek to mandate access to and scan encrypted content. They should think again."
  - Group Moderation Under End-to-End Encryption
  - Bugs in our pockets: the risks of client-side scanning (a blue ribon panel report)

---

## *XZ 1 — `liblzma` XZ backdoor to SSH (Easter Weekend)*

- Backdoor social-engineered into SSH upstream dependency
  - announced on what was Easter Weekend for many firms and countries
- Mostly affected rolling distributions (and unstable)
  - Some Fedora, Debian Sid, Kali, openSSE need(ed) to rollback.
  - x86/64 arch only (this time)
- Quite luckily found quite early due to serious benchmarking regression on an unstable test system;

- **AndresFreundTec** @AndresFreundTec@mastodon.social
- OpenWall OSS-Security
- CVE-2024-3094 NIST
  - XKCD:Dependencyexplained
  - @dfeldman@hachyderm.io
- Everything I know about the XZ backdoor (endorsed by Brian Krebs; by `@eb@social.coop`)
- ISC Podcast (Mon, Apr 1st): xz-utils Backdoor (CVE-2024-3094) ⇒ Diary 2024-04-01 by Bojan Zdrnja - The amazingly scary xz sshd backdoor

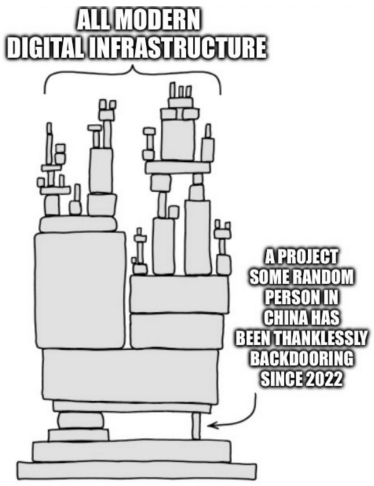
---

## XZ 2 — Backdoor inserted into SystemD dependency to attack SSH

- Why does SSH need XZ?
  - Because SystemD uses XZ compression, and some branches of OpenSSH integrate with SystemD.
- May have been differently lucky. SystemD had w.i.p to reduce dependencies including dropping this one
  - which may have caused attacker to get sloppy fast, causing the performance regression.
- "Junk drawer" libraries are valuable targets
  - especially if they have C++ initializers, code that runs upon being *loaded* not explicitly called.
  - So OOPS is an Ooopsie.

---

## XZ 3 — Hilarity Ensued …




---

## XZ 4 — Hilarity Ensued …

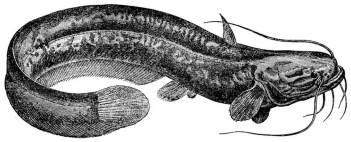

---

## XZ 5 — Hilarity Ensued …


---

## XZ 6 — Hilarity Ensued …

*reviving struggling projects the new way*



State sponsor your open source

*slowing down sshd like a boss*

O RLY? *Jia Tan*

## NOTES - XY, SystemD, and SSHD Backdoor

(Have I said before that SystemD is the Universal Attack Surface?)

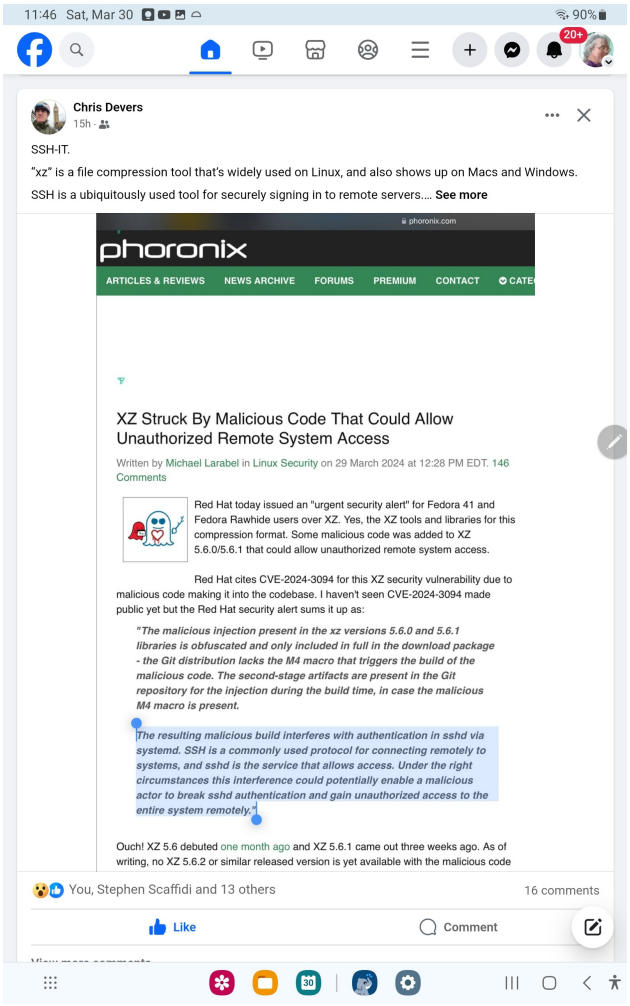
**AndresFreundTec** @AndresFreundTec@mastodon.social

> I accidentally found a security issue while benchmarking postgres changes.
>
> If you run debian testing, unstable or some other more “bleeding edge” distribution, I strongly recommend upgrading ASAP.

2/ > I was doing some micro-benchmarking at the time, needed to quiesce the system to reduce noise. Saw sshd processes were using a surprising amount of CPU, despite immediately failing because of wrong usernames etc. Profiled sshd, showing lots of cpu time in liblzma, with perf unable to attribute it to a symbol. Got suspicious. Recalled that I had seen an odd valgrind complaint in automated testing of postgres, a few weeks earlier, after package updates.

> Really required a lot of coincidences.

```
$ xz --version          ## simplistic, may check wrong one; 5.6.[01] bad.
xz (XZ Utils) 5.4.5
liblzma 5.4.5

 $ bash  detect_sh.bin   ## beter, checks which libzma sshd uses
probably not vulnerable

 $ bash -xv detect_sh.bin ## or, to trace it if paranoid, but only after
visual inspection

#! /bin/bash

set -eu
+ set -eu

 # find path to liblzma used by sshd
path="$(ldd $(which sshd) | grep liblzma | grep -o '/[^ ]*')"
++ grep liblzma
++ grep -o '/[^ ]*'
+++ which sshd
++ ldd /usr/sbin/sshd
+ path=/lib/x86_64-linux-gnu/liblzma.so.5

 # does it even exist?
if [ "$path" == "" ]
then
    echo probably not vulnerable
    exit
fi
+ '[' /lib/x86_64-linux-gnu/liblzma.so.5 == '' ']'

 # check for function signature
if hexdump -ve '1/1 "%.2x"' "$path" | grep -q
f30f1efa554889f54c89ce5389fb81e7000000804883ec28488954241848894c2410
then
    echo probably vulnerable
else
    echo probably not vulnerable
fi
+ hexdump -ve '1/1 "%.2x"' /lib/x86_64-linux-gnu/liblzma.so.5
+ grep -q f30f1efa554889f54c89ce5389fb81e7000000804883ec28488954241848894c2410
+ echo probably not vulnerable
probably not vulnerable

$
```




*phoronix*

[OpenWall OSS-Security]

> Date: Fri, 29 Mar 2024 08:51:26 -0700 From: **Andres Freund** <andres@…razel.de> To: oss-security@…ts.openwall.com Subject: backdoor in upstream xz/liblzma leading to ssh server compromise
>
> Hi,
>
> After observing a few odd symptoms around liblzma (part of the xz package) on Debian sid installations over the last weeks (logins with ssh taking a lot of CPU, valgrind errors) I figured out the answer:
>
> The upstream xz repository and the xz tarballs have been backdoored.
>
> At first I thought this was a compromise of debian's package, but it turns out to be

upstream.

…

((*includes attached analysis, detect.sh to check if compromize.*)

CVE-2024-3094 NIST

XKCD:Dependencyexplained

@dfeldman@hachyderm.io

> I hope the xz attack helps people understand: while a lot of sexy open source projects are supported by big corps or VC money, a lot of the unsexy ones are one or two people volunteering for a nearly-full-time level of effort and engagement, but for free.

Everything I know about the XZ backdoor (endorsed by Brian Krebs; by `@eb@social.coop`)

> Glyph @glyph@mastodon.social Mar 29, 2024, 20:43 I really hope that this causes an industry-wide reckoning with the common practice of letting your entire goddamn product rest on the shoulders of one overworked person having a slow mental health crisis without financially or operationally supporting them whatsoever. I want everyone who has an open source dependency to read this message https://www.mail-archive.com/xz-devel@tukaani.org/msg00567.html
>
> 2024-03-30 Joerg Jaspert :debian: @Ganneff@fulda.social And if you are curious about the #xz #compromise, a little update on the #Debian site: As already written, the archive processing is currently off (nothing new coming to testing/unstable/experimental, no mirror updates pushed out). Automated build daemons for the affected architectures have been stopped, and only two of them regenerated with a clean #stable environment. They are building for the security archive only, nothing else, right now. That part is safe. Members of the Release, FTP, Security, Build-Daemon and Sysadmin team are discussing what the next steps are. There are multiple different ways that can be taken, with different drawbacks and amounts of work involved. Also, it is not yet fully known what the malicious code all could do, so there might be much more that needs to be done later - or not. Unknown as of now, needs the analysis of it to finish, which is not easy nor fast. @debian
>
> analogist@social.ridetrans.it James Wu @analogist@social.ridetrans.it
>
> @grimalkina the part that stood out to me was that the decision from the poor maintainer to hand over the maintainership to the malicious actor was… expedited by a slew of psyops abuse from various sockpuppet accounts that were *absolutely indistinguishable* from the demands that OSS maintainers of popular packets get on a routine basis. e.g.: xz-devel@tukaani.org

> filippo@abyssdomain.expert Filippo Valsorda :go: @filippo@abyssdomain.expert 2024-03-30
>
> I'm watching some folks reverse engineer the xz backdoor, sharing some *preliminary* analysis with permission.
>
> The hooked `RSA_public_decrypt` verifies a signature on the server's host key by a fixed Ed448 key, and then passes a payload to `system()`.
>
> It's RCE, not auth bypass, and gated/unreplayable.
>
> More details in this thread: `https://bsky.app/profile/did:plc:x2nsu`

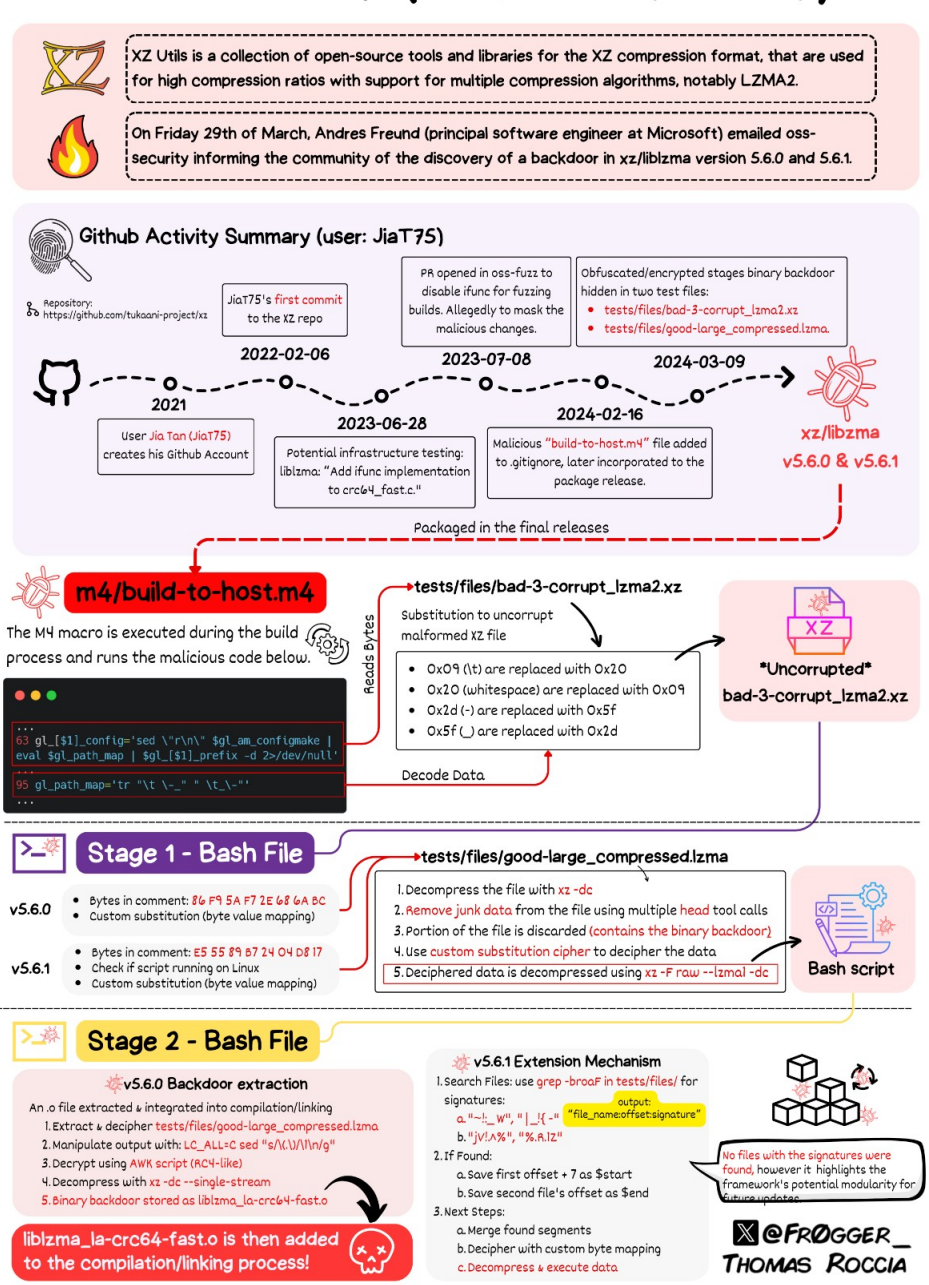
Jonathan Kamens jik@federate.social

> I want to point out the first item in @zackwhittaker@techcrunch.com's weekly #cybersecurity newsletter this week, as it pertains to the #xz backdoor. If you really think countries like China have engaged in "decade-long espionage campaigns" and haven't tried similar stunts elsewhere, you're kidding yourself. The xz backdoor isn't the only one; it's just the one we've found. #infosec Article link: https://www.justice.gov/usao-edny/pr/seven-hackers-associated-chinese-government-charged-computer-intrusions-targeting > **UK, US slap China's APT31 with sanctions** Newsletter signup: https://social.us18.list-manage.com/su

Kevin Beaumont For those asking what systemd change, easy write up: Reduce dependencies of libsystemd #32028

It was in train before the XZ issue was discovered, which may be why the threat actor sped up, started making mistakes and started begging distros to upgrade XZ - as what looks to be years of planning was about to be flushed down the pan.

Kevin Beaumont > I wrote a thing - Inside the failed attempt to backdoor SSH globally — that got caught by chance
Includes a few things which I suspect may be new insights.
https://doublepulsar.com/inside-the-failed-attempt-to-backdoor-ssh-globally-that-got-caught-by-chance-bbfe628fafdd
The level of sophistication of the XZ attack is very impressive! I tried to make sense of the analysis in a single page (which was quite complicated)!
I hope it helps to make sense of the information out there. Please treat the information "as is" while the analysis progresses! #infosec #xz




[Harald Hoyer @backslash@floss.social](#)

> Last Wednesday I gave a company internal Lunch&Learn talk about reproducible builds citing https://reproducible-builds.org/

> The motivation behind the Reproducible Builds project is therefore to allow verification that no vulnerabilities or backdoors have been introduced during this compilation process.

[reproducible-builds.org] > Reproducible Builds — a set of software development practices that create an independently-verifiable path from source to binary code > #xz #nix #nixos

benzblog: The XZ Backdoor via Fedi

excerpt -

> **Why does OpenSSH even depend on xz?** Normally, it doesn't, however several Linux distributions patch sshd to integrate systemd notifications. The systemd library to do that also happens to link against liblzma from the xz project. So perhaps the folks complaining about how systemd is so pervasive in modern Linux systems do have a point. … **Would SBOM have prevented this?** Haha lol no.

Tim Bray @timbray@cosocial.ca > 1/2 Looking at one of the #xz writeup, this struck my eye: > "The release tarballs upstream publishes don't have the same code that GitHub has. This is common in C projects so that downstream consumers don't need to remember how to run autotools and autoconf."
> Ah, GNU AutoHell, I remember it well. Tl;dr: With AutoHell, even if you're building for a 19-bit Multics variant from 1988, it's got your back. Except for it's just too hard to understand and use, thus the above. > #infosec

> 2/2 Thus, another #xz lesson. Don't rely on build tools you don't understand generally, and don't rely on GNU AutoHell specifically.
>
> [Yes, I understand this hack has many more moving parts, most much more sophisticated. But I didn't see anyone else saying this.]

Matt Blaze @mattblaze@federate.social

> An interesting thing about the XZ sabotage is that, while it was very cleverly obfuscated (congratulations to Andres Freund for finding it!), once found, it is very clear that it's a deliberate backdoor. It can't be explained away as an ordinary bug that introduced a vulnerability.
>
> Says something about the tradeoff space the attacker was working in.

2/ > Another notable thing: the backdoor, as far as I've been able to glean, is "NOBUS" ("nobody but us"), requiring knowledge of not just the existence of the backdoor, but also a secret key, to exploit. NOBUS backdoors would be very hard to camouflage as innocent bugs, so that could explain why the attacker chose this risky approach.

> But that raises the question of why they wanted a NOBUS backdoor. A state actor trying to limit colateral risks would likely care about this.

reply/

`Arik @arikb@mastodon.sdf.org`

> `@jripley @mattblaze` also, sshd drops privileges immediately after the authentication is successful. If you want to run something as root, it has to be pre-auth.

Dan Goodin ArsTechnica 4/1 updated Bruce

ISC Podcast (Mon, Apr 1st): xz-utils Backdoor (CVE-2024-3094) ⇒ Diary 2024-04-01 by Bojan Zdrnja - The amazingly scary xz sshd backdoor with Gist being updated & Jan Kopriva The xz-utils backdoor in security advisories by national CSIRTs - which countries managed to publish national advisories over a 3/4-day holiday weekend?

Filippo Valsorda @filippo.abyssdomain.expert 3/31 > To clarify, by "gated" I mean it takes the attacker's private key to use the backdoor (it's **NOBUS**); by "**unreplayable**" I mean that even if we observe an attack against one host, we can't reuse it against another host (the attacker's signature is bound to the host public key, but not to the command).

Krebs on OPSEC

Only affects x86/64 type architecture build, does not affect ARM/PI/…

"Junk drawer" libraries are valuable targets especially if they have C++ initializers, code that runs upon being *loaded* not explicitly called. So OOPS is an Ooopsie.

4/24 followup "Social engineering for open-source supply chain attack profit" Securilist - by GREAT - Global Research & Analysis Team, Kaspersky Lab

(*the irony of RU adjacent Kaspersky commenting on APTs*)

Schneier 4/2 XZ Utils Backdoor & 4/15 Backdoor in XZ Utils That Almost Happened

---

## *PuTTY Signature private key vulerable to key recovery compromise (April)*

- PuTTY 0.68—0.80 affected, including derivative graphic tools; fixed in PuTTY 0.81
- HN
  - CVE-2024-31497
  - PuTTY advisory
- Revoke any NIST-P521 keys used with Putty before the 0.81 update. (And maybe don't make new P521 either?)
- Root cause: attempt to generate nonces efficiently in virtualized environments resulted in sufficient bias the nonce fails to protect the private key.

- Moral: In Cryptology, efficiency is not the secret-keeper's friend.

## NOTES - PuTTY Signature private key vulerable to key recovery compromise

HN

> Following responsible disclosure, the issue has been addressed in PuTTY 0.81, FileZilla 3.67.0, WinSCP 6.3.3, and TortoiseGit 2.15.0.1. Users of TortoiseSVN are recommended to use Plink from the latest PuTTY 0.81 release when accessing an SVN repository via SSH until a patch becomes available.
>
> Specifically, it has been resolved by switching to the RFC 6979 technique for all DSA and ECDSA key types, abandoning its earlier method of deriving the nonce using a deterministic approach that, while avoiding the need for a source of high-quality randomness, was susceptible to biased nonces when using P-521.
>
> On top of that, ECDSA NIST-P521 keys used with any of the vulnerable components should be considered compromised and consequently revoked by removing them from ~/.ssh/authorized_keys files and their equivalents in other SSH servers.

CVE-2024-31497

PuTTY advisory

> Every version of the PuTTY tools from 0.68 to 0.80 inclusive has a critical vulnerability in the code that generates signatures from ECDSA private keys which use the NIST P521 curve. (PuTTY, or Pageant, generates a signature from a key when using it to authenticate you to an SSH server.) This vulnerability has been assigned CVE-2024-31497. It was discovered by Fabian Bäumer and Marcus Brinkmann of the Ruhr University Bochum; see their write-up on the oss-security mailing list.
>
> The bad news: the effect of the vulnerability is to compromise the private key. An attacker in possession of a few dozen signed messages and the public key has enough information to recover the private key, and then forge signatures as if they were from you, allowing them to (for instance) log in to any servers you use that key for. To obtain these signatures, an attacker need only briefly compromise any server you use the key to authenticate to, or momentarily gain access to a copy of Pageant holding the key. (However, these signatures are not exposed to passive eavesdroppers of SSH connections.)

? Therefore, if you have a key of this type, we recommend you revoke it immediately: remove the old public key from all OpenSSH authorized_keys files, and the equivalent in other SSH servers, so that a signature from the compromised key has no value any more. Then generate a new key pair to replace it.

(The problem is not with how the key was originally generated; it doesn't matter whether it came from PuTTYgen or somewhere else. What matters is whether it was ever used with PuTTY or Pageant.)

The good news: the only affected key type is 521-bit ECDSA. That is, a key that appears in Windows PuTTYgen with ecdsa-sha2-nistp521 at the start of the 'Key fingerprint' box, or is described as 'NIST p521' when loaded into Windows Pageant, or has an id starting ecdsa-sha2-nistp521 in the SSH protocol or the key file. Other sizes of ECDSA, and other key algorithms, are unaffected. In particular, Ed25519 is not affected.

**Details of the error:**

All DSA signature schemes require a random value to be invented during signing, known as the 'nonce' (cryptography jargon for a value used only once), or sometimes by the letter k. It's well known that if an attacker can guess the value of k you used, or find any two signatures you generated with the same k, then they can immediately recover your private key.

This means that it's dangerous to generate DSA signatures on systems with no high-quality source of randomness. Significantly more dangerous than generating the encryption keys for a single session: a leak of the private key compromises far more than one SSH session.

For this reason, since PuTTY was developed on Windows before it had any cryptographic random number generator at all, PuTTY has always generated its k using a deterministic method, avoiding the need for random numbers at all. The clever trick is to compute a secure hash whose input includes the message to be signed and also the private key. Secure hash output is indistinguishable from random data (or else the hash function isn't doing its job), and this generation method can't be repeated by an attacker who's trying to find out the private key – if they could generate the same hash input as you, they'd already have the private key.

This technique is now mainstream, and RFC 6979 documents a specific well-known way of doing it. But PuTTY didn't follow that specification, because we started doing the same thing in 2001, and the RFC wasn't published until 2013.

PuTTY's technique worked by making a SHA-512 hash, and then reducing it mod q, where q is the order of the group used in the DSA system. For integer DSA (for which PuTTY's technique was originally developed), q is about 160 bits; for elliptic-curve DSA (which came later) it has about the same number of bits as the curve modulus, so 256 or 384 or 521 bits for the NIST curves.

In all of those cases except P521, the bias introduced by reducing a 512-bit number mod q is negligible. But in the case of P521, where q has 521 bits (i.e. more than 512),

reducing a 512-bit number mod q has no effect at all – you get a value of k whose top 9 bits are always zero.

This bias is sufficient to allow a key recovery attack. It's less immediate than if an attacker knows all of k, but it turns out that if k has a biased distribution in this way, it's possible to aggregate information from multiple signatures and recover the private key eventually. Apparently the number of signatures required is around 60.

Stack exchange p521 curve

---

## *SSH breaks*

## Decades old attack still works on some SSHD implementations

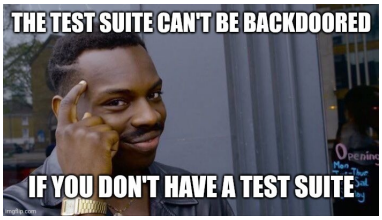
- A related general SSH attack: Old attack believed not to affect SSH will in some cases
  - poor choice of SSH settings, lack of mitigations allows use of old attack anyway by which signature computation failure can leak private key
  - RSA SSH keys only; 1 in 3 chance. yet another reason to phase out RSA keys -
  - ePrint, Ars, Schneier
- OpenSSH & OpenSSL implementations already had countermeasures, despite theoretically not needed just to be prudent, so not affected.
  - morale: always test a signature is verifiable before sending it
  - (and always verify a recieved signatures, and don't ignore failure!)
- older closed source e.g. embedded SSHd may be vulnerable?
  - especially if copy-pasta of example code instead of production quality code!

---

## regreSSHion - OpenSSH broken again


*or*, **Another Decades old OpenSSH server RCE remerges**

- 2024-07-01 Race Condition in OpenSSHD allowed Remote Code Execution due to skipped `if`
  - **regreSSHion** OpenSSH server remote code execution vulnerability
  - signal handler race condition; critical `if`-statement bypassed in latest patches
- bug fixed in OpenSSH 4.4p1 (2006) un-fixed in 8.5p1 (2020), re-fixed 9.8p1 (2024).
  - CVE-2024-6387 (regression of CVE-2006-5051)
  - reported as glibc-based Linux OpenSSHd, and everyone patched. but
- "The Windows OS bundled version of OpenSSH appears to be vulnerable to CVE-2024-



6387 aka regreSSHion - it is version 8.6.0.1." (Kevin Beaumont)
    - "The Windows OS bundled version of OpenSSH appears to be vulnerable to CVE-2024-6387 aka regreSSHion - it is version 8.6.0.1."
        - Microsoft Win**11** PowerShell Issue

---

### *Telegram pot calls Signal kettle black, film at 11*

Telegram MTProto 1.0 remains vulnerable to Chosen Ciphertext attack, and default is not end-to-end so subpeonable?

- It's Time for Furries to Stop Using Telegram,
    - Soatok notes their chosen Telegram username is a protest `IND_CCA3_Insecure`
        - because Telegram MTProto 1.0 fails under `Indistinguishability under chosen ciphertext` attacks;
        - 2.0 status under discussion
        - Details
- Matthew D Green's X thread followed Elongated Muskrat + Telegram's attack on Signal Protocol (Signal/WhatsApp/etc), seemingly to drive activists from Signal to "mostly unencrypted Telegram".
    - Matthew Green's blogpost: "Is Telegram really an encrypted messaging app?"
    - Kaspersky

## NOTES

*The irony of citing a good explanation from Kaspersky but …*

---

## *“Unpatchable vulnerability in Apple chip leaks secret encryption keys”*

Fixing newly discovered side channel will likely take a major toll on performance.

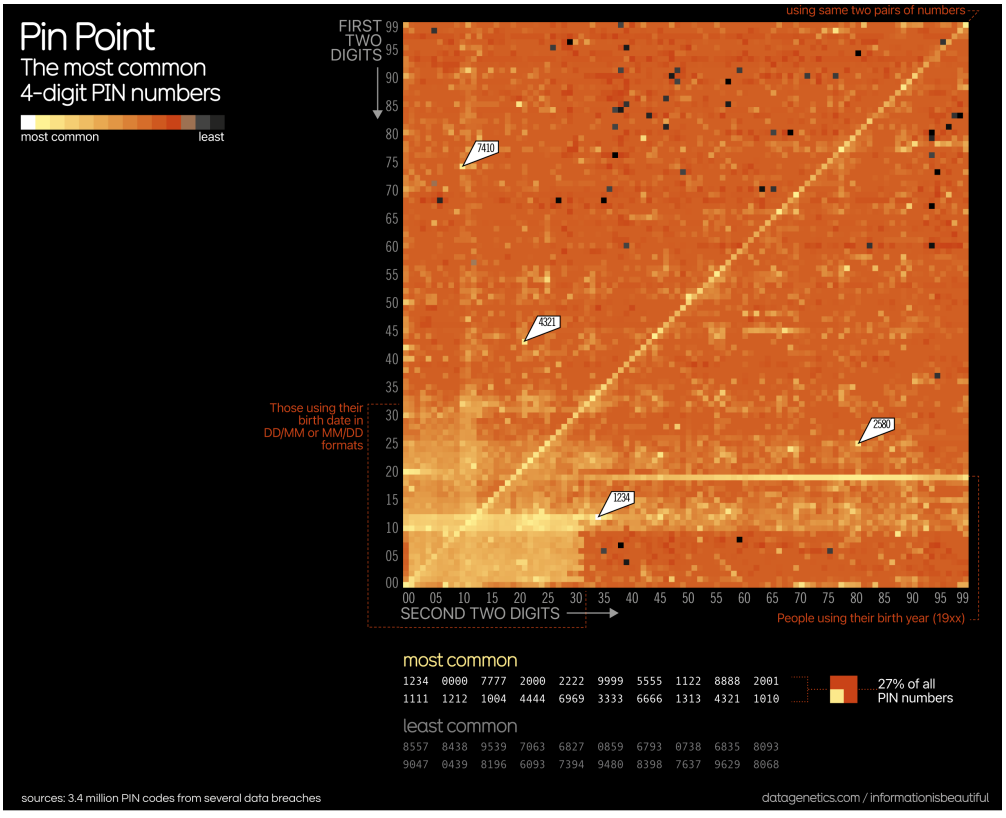
Dan Goodin - 3/21/2024 Ars

“Beware of hardware optimizations”

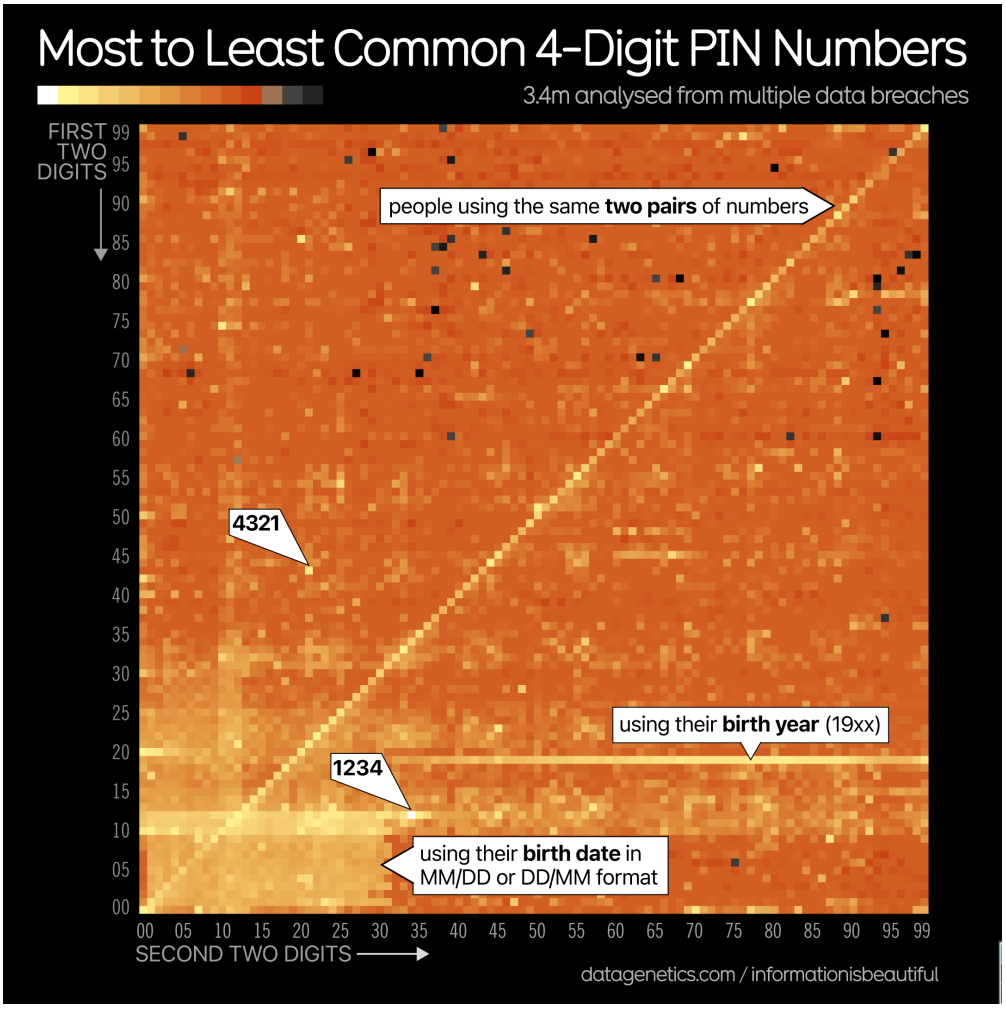
see also Schneier 3/28

## *Bad 4 digit PINs*

**A Study of 4 Digit PIN popularity**, inspired by humor, using real leaked PIN collections, briefly went viral.



## *Bad 4 digit PINs (continued)*




*4 Digit Pins*

## NOTES

Well, a 4 digit limit on PINs is pretty bad in itself. Limiting to digits is required if to be entered on a phone pad or equivalent; and requires Denial of Service lockout to prevent bruteforcing all 10k.

Study is over a decade old (tagged 2009-2013) but went viral in 2024, so I'm including it in 2024 news.

As author Nick comments, do not use his list of 20 least used PINs, the bad guys will have already grabbed that list assuming you'd do that.

How viral was this? You probably saw it in passing.
This story was discovered and promoted by **Information is Beautiful** as Most Common PIN

Codes; they syndicate widely, so viral it went.

- Wired
- TwitTV SN! 13' {who alas credits Information is Beautiful who pushed it viral, not the data engineer}.
- Reddit … Twitter … FB … Mastodon/Fediverse … pretty much everywhere

In the original article, if one scrolls down, he extends his study to all numeric pass-codes of any length. Half are 4 digit. 6 and 8 digit pass-codes account for another 30%. The remaining 20% are a not-so-fat tail 5,7,9,10..14. 123456… is tops in each length. Geometric patterns on the keypad remain popular, same as they were with German ENIGMA operators early in WW2.

Similar data, four-digit-pin-codes-sorted-by-frequency-withcount.csv

Alas the data scientist who did this wonder, **Nick Berry**, succumbed to cancer in 2022.

---

## *YubiKey Sidechannel (September)*

YubiKey Sidechannel Attack

- sidechannel leaks sufficient information to clone a key.
  - this requires a YubiKey to be inserted into a hostile machine
  - e.g. portable device of industrial spy who finds YubiKey on your desk, or public library public access terminal
- so
  - do not leave unattended!
  - use a strong PIN he says
- NinjaLabs
- Yubico advisor 2024-03
- Ars

---

## *Blockchain garbage collection (fall 2013, retro publ. Jan 2014)*

**Blockchain garbage collection** `{paywall}`

**How a 27-Year-Old Codebreaker Busted the Myth of Bitcoin's Anonymity.**

> Once, drug dealers and money launderers saw cryptocurrency as perfectly untraceable. Then a grad student named Sarah Meiklejohn proved them all wrong—and set the stage for a decade-long crackdown.

- YT
- Dr Sarah Meiklejohn now (full) Professor of Cryptography and Security at UCL

Key sentence of abstract

> Bitcoin has the unintuitive property that while the ownership of money is implicitly anonymous, its flow is globally visible.

TL;DR Built a mesh or map using Metadata of transactions to cluster public and supposedly private wallet IDs of inferred same actors.

## NOTES

I feel unclean linking even a16z's Y-T.

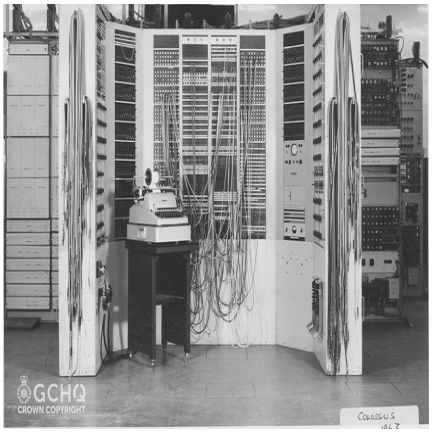
Dr Meiklejohn's academic paper evokes Clint Eastwood two movie titles "**A Fistful of Bitcoins**: Characterizing Payments Among **Men With No Names**."

Wikipedia reference links include Wired 2024, NYT 2013, MIT Tech Review 2019.

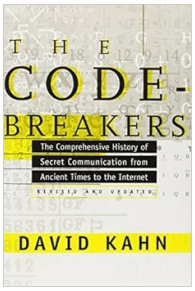
---

## *Potpourri*

- **Windows** belatedly eliminating **RSA1024** *internal* keys via SANS ISC Microsoft announced deprecation of 1024 bit RSA Keys
- Recent work on **Voynich Manuscript** The Atlantic c/o Schneier's summary
- **Protocols**, **Implementations**, and **People** are still the Achilles Heal of most implementations
  - 2024-07-10 Blast-RADIUS MITM attack on RADIUS authentication protocol Ars
  - Above mentioned race-condition due to code regression failure.
  - saving private credentials in published source-code (either coded in-line, or failure to `.git-ignore` the credential files in project dir)
  - XZ malware was social engineering
- **Colossus**:
  - New Images of Colossus released for 80th Anniversary
    - Ars
    - GCHQ official
  - Dollis Hill War Diary declassified;
    - Museum Crush ;
    - BT official;
    - Trent Park House of Secrets




- **David Kahn (1930-2024)**
  - funeral;
  - WaPo,
  - Schneir,
  - Cryptologia 48:3




---

# §2 What's up with Post Quantum Cryptography?

## *Review: What's Quantum Computing?*


(Review of 2022 status statements, followed by 2024 updates tagged **2024**.)

Quantum Superposition when used for computing.

- QC measured in "**qubits**" not bits
- 30% True, 70% False.

  Such bits are in quantum superposition of True and False, which is a *bug* in classical computing but a *feature* in QC.

  This allows non-deterministic algorithms.

## Review: Kinds of Quantum Hardware

- Quantum Annealing - big qubit counts, great for optimization problems

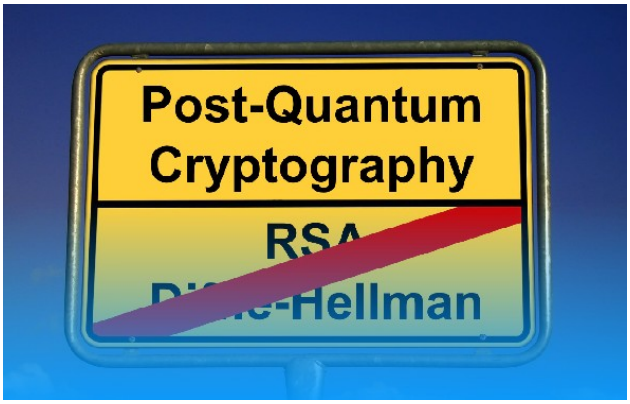
  **but not cryptology.** (?yet? well maybe …) *Not general purpose.*

- Quantum Circuit/Logic - small numbers of qubits so far.

  In theory, algorithms for these hardware types can use non-deterministic parallelism to evade classical performance limits, and in particular, could allow factoring fast enough to be dangerous, provided big enough quantum circuits can be made to work.

## Review: We're discussing PQC before QC?

Yes !




- **Quantum Cryptography**
  - theoretically using entangled quantum states
  - to create an encryption
  - or an anti-eaves-droppable connection
- Quantum Crypt**analysis**
  - Using Quantum Computing to defeat classical PKI encryption
- **Post Quantum Cryptography**
  - new classical encryptions that can resist Quantum Cryptanalysis,
  - so read as Ready for post-**(**Quantum-Computing**)** Cryptography.

## Review: What's the problem?

- Unbreakable ciphers aren't always unbreakable, for always.
- QC *could theoretically* break most PKI
  - Schor's Algorithm / Grover's / VQF
  - discrete log as well as prime factoring, even elliptic curves
- **2024-01-05** Improving Shor's Algorithm
  - "Thirty Years Later, a Speed Boost for Quantum Factoring" (Quanta)
  - Oded Regev's paper on arXive
  - and another team already reduced the memory penalty of the speed improvement arXiv

## NOTES

Every unbreakable cipher has been broken eventually (at least partially[1]).

20thC RSA and other PKI not guaranteed proof against either of:

- major breakthrough in number theory (factoring &/or discrete log), or
- quantum hardware + algorithms (practical fast factoring )

Schor's Algorithm in theory would factor fast on enough quantum circuits but 21 is not a large

1 See our prior discussions of GEE, VENONA for breaks of One Time Pad

number yet. (see also Wikipedia. Some say 433 bits on IBM Osprey QC is enough for RSA2048 with Schor's algo needing 372 Qubits (with pre-processing and post-processing), but will it work? Schneier and Schor doubt it. *Shouldn't someone try it?*)

Other probabilistic quantum algorithms (Grover, GEECM, Variational Quantum Factoring (VQF)) can do *some* much bigger numbers (*which may just define new class of unsafe primes??*), and with classical pre-processing, can use a much smaller number of qubits than the ^*obvious*^ $\log_2 N$.

*not clear this will ever be able to generally break RSA4096, but it's not impossible, so prudent to plan for that day.*

---

## Review: Generalization of Forward Secrecy

- Classical "Forward Secrecy" - old messages not broken by later loss of host key
- Generalized: old saved messages not broken by later breakthroughs either.
- Realistic threat?
  - VENONA + GEE 1940s {BLU Sept 2018}
  - NSA Utah Data farm, 2013


## NOTES

```
* VENONA: It worked Once! {[BLU Sept 2018]
(http://blu.org/meetings/2018/09/)}
* We now have a Vacuum Cleaner of Holding (_Greenpeace photo c/o Wikimedia_)
```

So yes, it **can** happen again.

Normal Forward Secrecy requires that if e.g. the Host Key is compromised later, any retained cryptograms sent with nonce keys negotiated with the compromised Host Key aren't also compromised.

This is nice, but we'd also like to protect against advances of technology, e.g. fast factoring or solutions of discrete logs.

This may not be within *your* threat model, yet, but in dystopian plausible futures, things you've already discussed/downloaded might be retroactively illegal/disloyal and oops.

---

## *Review: NIST's Post-Quantum Cryptography Standards*

> The goal of post-quantum cryptography (also called quantum-resistant cryptography) is to develop cryptographic systems that are secure against both quantum and classical computers, and can interoperate with existing communications protocols and networks. –

**NIST**

## Review: NIST PQC Competition

National Institute of Standards & Technology started a multi-round competition, similar to with AES and SHA3 competitions

- NIST announcement

- NIST Q&A
- Schneier on PQC

## NOTES

NIST, the Bureaucracy formerly known as NBS.

*Goal is to have PQC ready for use not only **before** quantum breakthrough but early enough (roughly now) that anyone who wishes to avoid save-intercepts-now-to-break later can switch quickly; although it may already be too late WRTO NSA archive?*

*This competition was "more brutal" than prior; of 69 candidates, peer cryptanalysis has broken 62. So far.*

---

## *Review: Quantum Cracking 2023*

- **RSA2048 in play or not?** - Chinese academic paper claiming 2k bit RSA within range of current gen NON-fault-tolerant QC, no great surprise given Qubits available and theoretical algorithm size. Schor and Schneier unconvinced - does it actually converge w/o FT? Schneier 2023-01

- Schneier "You Can't Rush PQC Standards"

- Quantum resistant hybrid-signing FIDO2 keys for 2FA

- **Side-Channel Attack against CRYSTALS-Kyber**

[2023.02.28] CRYSTALS-Kyber is one of the public-key algorithms currently recommended by NIST as part of its post-quantum cryptography standardization process. Researchers have just published a side-channel attack—using power consumption—against an implementation of the algorithm that was supposed to be resistant against that sort of attack. The algorithm is not "broken" or "cracked"—despite headlines to the contrary—this is just a side-channel attack. What makes this work really interesting is that the researchers used a machine-learning model to train the system to exploit the side channel.

OTOH as seen in TETRA:BURST, a side-channel attack can be used to extract key or algorithm from a piece of equipment that falls into opponent lab.

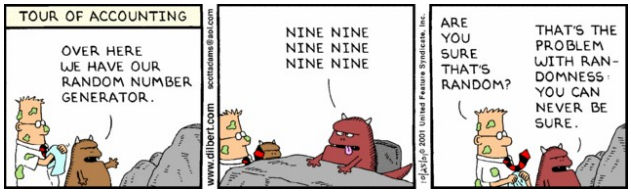
---

## REVIEW: Known weaknesses

- breaks eliminated 62 of 69 entrants in Rounds 1 to 4
- including the two front-runners, Rainbow and SIKE
- 7 remain, will they survive?
- FALCON would be compromised by a lack-of-randomness in salt, or failure to salt, as repeating same key and hash again gives too much information.

## Isn't non-random or uniformly-blank Salt an unlikely failure?

TL;DR ***No. It's happened.*** (*see in notes*)




## NOTES

Lack of randomness failure isn't just hypothetical, lots of SSH keys got invalidated in 2008 because they were well-known-primes.

- Numerous implementations have failed to salt encryption of small data despite warnings.
- DEBIAN broke system `random`^[`DSA-1571-1 openssl` predictable random number generator {CVE-2008-0166} {Schneier} which compromised many SSH keys.
- our historical vignette in prior years has discussed danger of key reuse in WW2 and Cold War. Lack of Salt = Key Reuse.

(**WTAF**? Yep. Debian packagers applying *normal best practices* where they shouldn't even touch (Normal doesn't apply!) had *removed* the entropy-harvesting because Valgrind and Purify gave `accessing uninitialized memory` warnings. Well yeah, *that's how we harvest entropy*! Another problem (mostly solved?) is host key generation at VM start - the VM's entropy is rather deterministic (biased) at that point. Similarly, optimizing compilers removing zeroing memory prior to releasing it can allow keys to leak into the memory pool. Cryptographic software is an ongoing a battle against computer ^science^ that ^knows better^.)

And failure to salt wouldn't surprise me when non-specialists (applications developers, database programmers, protocol developers) who should stick to packaged PKI use-case libraries (e.g. **NaCl**) try to use cryptographic primitive routines directly to avoid dependencies.)

2023 added few more low-entropy initialization examples added to the list.

And 2024's PuTTY key disclosure was due to implementing low-entropy nonces badly for use in VMs and everywhere else.

*Won't someone think of the random numbers?*

---

## *NIST PQC Timeline*

- 2022-04-28 How to Prepare Your PKI for Quantum Computing (April 28, 2022)
- 2023-03-03 ✓ Post-Quantum Cryptography Conference (Friday March 3, 2023 - Ottawa, Canada)
- 2023-03-16 (podcast) Root Causes 286: PKI and PQC in New White House Cybersecurity Initiative (Mar 16, 2023)
- 2023-08 ✓ FRN RFC Draft Standards FIPS 203, 204, 205.
- **2024-04** ✓ Fifth PQC Standardization Conference
- **2024-08-15** ✓ FIPS Standards; `FIPS Allowed`: NIST announced finalized PQC standards for 3 of 4 "winners" (3 more to come)
- 2025/26 FIPS certification for the PQC algorithms; `FIPS Approved`.

NIST PQC

## 2024-08-15 NIST Releases First Post-Quantum Encryption Algorithms

- As expected, on schedule
- FIPS 203, 204, 205
  - PQC key-encapsulation **CRYSTALS-KYBER** (ML-KEM)
  - two+ PQC Signature schemes CRYSTALS-Dilithium (ML-DSA), SPHINCS+ (SLH-DSA).
    - third PQC Signature FALCON (FN-DSA) finalization still w.i.p due this year
  - two other sets also w.i.p as backups.
- NIST
- Schneier

## *2024: Quantum Computing Export Control Conspiracy?*

- 2024-07-03 NewScientist: Multiple nations enact mysterious export controls on quantum computers
  - "Identical wording placing limits on the export of quantum computers has appeared in regulations across the globe. There doesn't seem to be any scientific reason for the controls, and all can be traced to secret international discussions[.]"
  - Matthew Green: "I'm curious whether this is a case of "national security types without enough information getting panicked" or if there's any substance behind this."

---

## *2024: Quantum Algorithm News*

- 2024-04-10 Quantum Attack on PQC Lattice underpinnings announced
- 2024-04-19 … and fails, rapidly

    - Chen's paper "Quantum Algorithms for Lattice Problems"
    - claimed "polynomial time quantum algorithm for solving LWE"
    - Algorithm in paper has a bug, unclear if fixable, so danger averted *for now*
        - but as we say "*Attacks only get Better*"
    - commentary — Matthew Green; Schneier weeks later
- 2023-12-22 IEEE Spectrum: "**Quantum Computing's Hard, Cold Reality Check**"
    - "Hype is everywhere, skeptics say, and practical applications are still far away"
    - Spectrum ; Schneier response
- 2023-10-06 & 11-30 Matthew Green essay To Schnorr and beyond (Part 1) & (Part 2) discussing how signature protocols might work with PQC Dilithium algorithms.

## Notes - Quantum Algorithm

- Chen's Quantum algorithm is an attack on the Lattice maths that NIST PQC is largely built upon, so this would be significant
    - if it wasn't broken.
- All Quantum attacks on cryplography presumes
    - QC HW scales up and maintains (fault-tolerance?) coherent superpositions long enough to perform useful calculations
    - which is not demonstrated yet
    - but prudent for NIST and Industry to plan our migration before disaster is upon us

---

## *2024: Chinese researchers Claim RSA, AES encryption break with a commodity quantum computer (1)*

"*If it sounds too good to be true…*"

Claim is break of "military grade" RSA and symmetric algorithms (*like* AES) using the cheaper/simpler D-Wave Quantum Annealing computers.

"*Interesting if true*" as Cousin Millie taught us to say.

- 2024-10-14 MONDAY FUD at CSOonline
- Everyone except SCMP links an older paper!
    - with only an English abstract, rest Chinese;
- May 2024 paper claims factoring 50bit RSA-ish semiprime
    - "And we implemented the first 50-bit integer decomposition on D-Wave Advantage."
        - n = 845546611823483 = (p×q) = 40052303 × 21111061 is correct.
        - 50bit is impressive only because this is on D-Wave Quantum Annealing, not *per se* Quantum Computing (QC)
    - their algorithm for factoring a 50bit semiprime with QAnnealing is highly unlikely to scale to RSA2048, RSA4096.

---

## 2024: RSA-ish small integer check

```
$ perl -Mbigint -E 'my $n=845546611823483; my $p=40052303; my $q=21111061; say
($p*$q); say sprintf(q(%d x%x (%db)),$_,$_,length(sprintf(q(%b),$_))) for ($p,
$q, $n); say $n-($p*$q);'
845546611823483
40052303 x263264f (26b)
21111061 x1422115 (25b)
845546611823483 x301052970537b (50b)
```

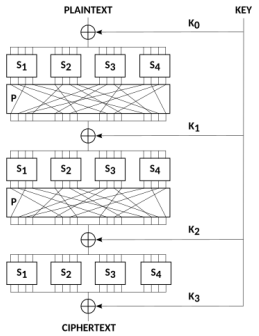
If they actually had **scalable** *factoring* with D-Wave Quantum Annealing, that would be shocking breakthrough. That is ^known^ to require real QC like IBM's; factoring interseting numbers isn't believed within D-Wave's capability. Their demonstration of 50b does not appear to be a breakthrough.

---

## 2024 October Surprise: Chinese researchers Claim RSA, AES encryption break with a commodity quantum computer (2)

- South China Morning Post earlier this week SCMP {paywall}

  Chinese scientists have mounted what they say is the world's first effective attack on a widely used encryption method using a quantum computer. The breakthrough poses a "real and substantial threat" to the long-standing password-protection mechanism employed across critical sectors, including banking and the military, according to the researchers. Despite the slow progress in general-purpose quantum computing, which currently poses no threat to modern cryptography, scientists have been exploring various attack approaches on specialised quantum computers.




- The claim is that they can solve 3 modern "Military Grade" SNP block-ciphers is frankly far more interesting than a non-scalable solution to toy RSA integers.
  - SNP := Substitution/Permutation Network
  - But not backed by the facts seen so far.
  - Full rounds or reduced? Only Lightweight ciphers, not AES *per se*.

## Notes on this week's D-Wave alleged break of symmetric ciphers

All analysis is preliminary as most of us haven't seen the paper yet …

- If one can crack full set of rounds of a major SNP cipher used for actual payloads, single cryptotext, no cribs, that would be big.
  - Then breaking RSA used for session/message key negotiation would not be needed for Evesdroppers to read the Text
  - Factoring would still be needed to forge signatures

    - However, claim is AES-like not AES (GIFT-64, RECTANGLE, PRESENT)
- Coverage so far
    - Tom's Hardware
    - NPR
    - ElReg
    - TechTarget
    - PCMag
    - My BSKY Thread

---

# §3 History Vignette - Breaking the Silk Dress Cryptogram

A Coded Message found in vintage victorian dress

## *Breaking the Silk Dress Cryptogram*

A cryptogram was found in the pocket of a 19thC silk dress.

Solution required understanding how synoptic meteorologic observations were collected done by telegraphy.

*Published academically in August 2023, but released as a Christmas story echo Dec. 2023.*

- UManitoba
  - MP4 17'
  - PDF

## *Notes for Silk Dress*

(*No "Bennett" was found in Weather Bureau empolyees. Was she possibly a commercial or US Army Signal Service civil Telegraph employee in Wash.D.C. who took a personal copy of two collated messages?*)

- **Additional Video for Silk Dress Solution**
  - Video of finder **Sara Rivers Cofield** showing the silk bustle dress `Bustle_dress_pocket4.mp4 (170M)` {alas original deleted}
    - MSN 2'17"
    - lecture 1:03:38
    - TV 3'23"
    - CTV 2'19"
  - Video of solution / solver and Sara
    - CBC TV 2'21"
  - Video of others, in context of solving unsolved messages
    - "**More Famous and Not-So-Famous Unsolved Codes**" - Unsolved Codes Talk - Discussion of the Silk Dress cryptogram by historical cryptologists Klaus Schmeh and Elonka Dunin, as part of a virtual prestation hosted by the National Museum of Computing at Bletchley Park, UK (April 18, 2021; before solution). Y-T
    - Unsolved Ciphers and… there's an explanation of the Silk Dress cryptogram as just one part of a video that goes through an "iceberg" chart of unsolved codes and ciphers (one of **many** in part 2). User `TesseractHeart @tesseracthearts`; Iceberg playlist parts 1 & 2
- **Academic and Professional**
  - **Breaking the Silk Dress Cryptogram** (Research Article; *Cryptologia* 2023-08-06) preprint
  - NOAA — 'Cryptogram' in a silk dress tells a weather story
  - Conservator Sara Rivers Cofield on conservation (not the cryptograph) Y-T

1:03:37

- **UManitoba's cached resources**
  - link to same 17-minute graphic novelization of the solution
  - UMan resource page ⇒ Sara Rivers Cofield's "Commitment to Costumes" blog post about the Silk Dress cryptogram
  - Cipher Mysteries blog: Blog post about Silk Dress cryptogram on cryptologist Nick Pelling's Cipher Mysteries website.
    - ⇒ CipherMysteries
  - 135-Year Old Code Mystery Highlights Forgotten Era of Meteorology
  - Research Article
- **Popular**
  - NYPost
  - The Weather Network

---

# §4 Bibliography & Footnotes

## *My talks*

The **YouTube** of this presentation will be linked on BLU.org along with these slides and extended notes *etc* as 2024-oct as per usual.

**Prior talks in this series** - most talks have slides &/or YouTube attached, sometimes extras.
*Alas the YouTube audio pre-pandemic wasn't great, BLU will need a donation of a wireless clip-on mike if we ever return to Hybrid/In-Person meetings. Or we all need to wear a wired or BT headset while presenting in person?*

## *News + Focus*

**News** and **Focus** sections have embedded links.

Good security news streams to either research history or to follow year round are Scneier Crypto-gram and SANS ISC, the latter being less cryptologic and more operational in focus – but both cover the wide span of vulnerabilities, tools, remediations, etc, not just the cryptologic that I'm cherry-picking here.
**Highly recommended**.
*Start your day with the 5 minute SANS Internet Storm Center StormCast pod-cast; the Red Team is, so, so should you.*

## *Cryptologic History - general references*

- ***Bletchley Park Podcast*** on your favorite pod server
  - FB
  - BPP Keyword index
    *by me, current thru fall '22*
- **Books**
  - *The Code Breakers*, revised & updated; Kahn, David; 1996: NY S&S.
  - *Decrypted Secrets*; Bauer, F.L.; 1997: Heidelberg, Springer.
  - Schneier books
- **Websites**
  - Wikipedia
  - Cipher History
  - CryptoMuseum
  - NSA History
  - GWU NSArchive (Academic National Security Archive at GWU; FOIA Declassified)
  - Declassified post-WW2 **TICOM** reports (*Signals and Cryptologic Intelligence equivalent of better-known Operation Paperclip, etc.*)