Post-Quantum, Measured

Migrating TLS, PKI, and application crypto with evidence you can reproduce

Danny B. Carr, Jr.

Table of contents

Post-Quantum, Measured

Migrating TLS, PKI, and application crypto with evidence you can reproduce

Danny B. Carr, Jr.

A hands-on, production-minded guide to migrating real systems to post-quantum cryptography, written from the practitioner's chair rather than the lecture hall. It follows the NIST standards (FIPS 203/204/205) and the deployment reality behind them: what the algorithms cost, how to move TLS and PKI, what auditors ask, and how a live zero-knowledge platform actually shipped a hybrid post-quantum migration.

The production case studies, including the Envelope v3 post-quantum migration, are drawn from **Etergis**, a shipped, verified codebase, not a simulation built for the book.

> **Verified / Reported / Proposed.** Every technical claim is labeled. *Verified* means measured or run in the accompanying lab with evidence in the repository. *Reported* means sourced from a published standard or paper, cited inline. *Proposed* means a design that has not yet shipped or been measured, called out so it is never mistaken for a result. See the Preface and Front Matter for the full reading guide.

Use the sidebar to navigate, the search box to jump to any topic, and the **PDF** / **EPUB** links in the sidebar to take the book offline.

Post-Quantum, Measured

Danny B. Carr, Jr.

Copyright & License

© 2026 Danny B. Carr, Jr. All rights reserved.

Published by Carr Digital LLC.

Buyers receive every subsequent revision at no additional cost. Manuscript text is not licensed for redistribution, resale, or use as training data without written permission.

The code listings and the captured lab output that back every claim marked **Verified** are in the companion repository at `github.com/DannyBCarrJr/post-quantum-measured-lab`, released under the MIT License. Run them. If a Verified number in this book does not reproduce from that repository on OpenSSL 3.5 or later, that is a defect worth reporting.

Trademarks and product names belong to their respective owners and are used here for identification only.

About the Author

Danny B. Carr, Jr. is the founder of Carr Digital LLC, where he builds and operates Etergis, a zero-knowledge digital continuity platform. The production case studies in this book, including the Envelope v3 post-quantum migration, are drawn from that platform's real, shipped, and verified code. They are not a simulation built for the book.

He works in enterprise security engineering, with a focus on public key infrastructure and multi-factor authentication: issuing, monitoring, renewing, and revoking certificates, and automating the resulting toil in PowerShell and Python. Keeping a trust chain unbroken and an audit log legible is the practitioner's-eye view this book is written from.

Before he specialized in PKI he taught cybersecurity bootcamps covering ethical hacking, network security, and vulnerability assessment. Before that he spent two decades in law enforcement and public safety. He holds the CompTIA Security+, CySA+, and CASP+ certifications and is working toward the CISSP.

This is a personal work, researched and written entirely on the author's own time and personal equipment. All views, opinions, and recommendations are the author's alone. They are neither made on behalf of, nor endorsed by, any current or former employer. Every technical claim derives from publicly available standards and documentation, or from the author's own reproducible home lab. No confidential, proprietary, or employer-owned information, data, or intellectual property is included.

How to Read This Book

This is a hands-on, production-minded book, not an academic one. A few conventions carry through every chapter:

Verified / Reported / Proposed. Every technical claim in this book is stamped with how it is known. Nothing here is merely asserted.

- **Verified:** measured or run in the companion lab, with the evidence checked into the repository's `evidence/` directory. If a chapter states a number without qualification, it is this category.
- **Reported:** sourced from a published paper, standard, or vendor document, cited inline. Not independently re-derived in this lab, but traceable to its source.
- **Proposed:** a design or recommendation that has not yet shipped or been measured. Called out explicitly so it is never mistaken for a verified result.

The distinction exists because a "the migration is zero-risk" claim that turns out to be an untested assumption is worse than useless. It actively misleads the one reader who trusted it enough to act. The companion repository states the same policy, and every Verified claim in this book has its script and its captured output sitting there for you to re-run.

Code you can run. Every lab referenced in a chapter has a real script in `src/`, and real output captured in `evidence/`. The labs assume Ubuntu 26.04 LTS, OpenSSL 3.5.5+, and Python 3.14. See the Preface and Appendix D for the full environment and how to reproduce it on an earlier distribution.

Chapter structure. Each chapter opens with a concrete hook (a failure, a deadline, a measured surprise), states what it covers, then works through the material with labs and listings inline. Chapters close with a short prose bridge into what comes next, not a bullet-recap of every heading.

Preface

Why Now

The question I am most often asked is "why do I need to worry about this now if quantum computers don't exist yet?"

The answer is lead time.

A root CA submitted to browser trust programs today may be accepted in two to five years. A PKI re-keying project takes 12 to 24 months minimum. An application-layer encryption migration requires design, implementation, testing, and staged rollout. The data you encrypt today may still need to be protected in 2035. Adversaries collecting traffic now are counting on you waiting.

The algorithms are standardized. The implementations are in stock OpenSSL. The deployment is a configuration change for TLS key exchange. The only missing ingredient is starting.

Who This Book Is For

This book is for security engineers who deploy and maintain cryptographic systems: PKI administrators, platform engineers, application developers, and security architects who need to understand post-quantum cryptography well enough to make real decisions and implement real changes.

It is not a textbook. You will not find proofs here. You will find working code, verified benchmark numbers, production-relevant gotchas, and a clear path from "I've heard about PQC" to "I've deployed X25519MLKEM768 and my team understands why."

What This Book Assumes

You are comfortable with TLS, certificates, and public-key cryptography at an operational level. You know what a CA is, what ECDH does, and why forward secrecy matters. You have access to a Linux terminal.

You do not need a mathematics background. The algorithm descriptions in this book are deliberately intuition-first. Where the math is required for understanding, it is explained in plain language. Where it is not required for practical deployment, it is omitted.

The Lab Environment

Every code listing in this book was written and verified in this environment:

```
OS:       Ubuntu 26.04 LTS (Plucky Puffin)
OpenSSL:  3.5.5 (native FIPS 203/204/205 in default provider)
Python:   3.14.4
Hardware: Intel i7-13700HX, 64 GB RAM
```

Critical: OpenSSL 3.5.0 is the minimum version. Earlier versions require the OQS provider extension for PQC algorithms. Ubuntu 26.04 ships 3.5.5 by default. For earlier Ubuntu versions, see Appendix D for installation instructions.

No additional packages, providers, or compiled extensions are required. If you have a terminal and OpenSSL 3.5.x, every lab in this book runs as written.

The Lab Repository

All lab code, evidence artifacts, and benchmark data are in the companion repository: `github.com/DannyBCarrJr/post-quantum-measured-lab`. The evidence directory contains the actual output of every lab run, not fabricated and not adjusted. If a command produces different output on your hardware, that is expected: different processor, different OpenSSL build. The algorithm parameters (key sizes, ciphertext sizes, signature sizes) are deterministic and will match exactly.

A Note on Verified Claims

Every technical claim in this book is stamped with how it is known, using the three labels defined in the front matter. **Verified** means I measured it here, and the code and the captured output ship with the book. **Reported** means it traces to a FIPS specification, a published paper, or a vendor document, cited inline, but I did not re-derive it. **Proposed** means a design that has not yet shipped or been measured, and it says so rather than hiding in the prose.

The labels exist because of one specific bad day. In early July 2026 I found two labs in my own repository, both presenting themselves as "the Etergis Envelope v3 design," quietly using different constructions: their HKDF domain-separation strings didn't match. The same day I caught the migration chapter describing a rollout as "zero breaking change" when what had actually been demonstrated was a wire-format property; the rollout itself was unimplemented, and the claim contradicted the platform's own history, which includes a hard break during beta. Neither was malicious. Both were drafts drifting ahead of the evidence, which is how confident wrong claims are usually born. The Verification Standard went into the repository that day, and every claim in this book has carried one of the three labels since.

Opinions and recommendations are labelled as opinions and recommendations. Claims about vendor support and regulatory status were accurate as of mid-2026, and are flagged where they are the kind of thing that moves.

If you find an error, a better approach, or a claim that doesn't hold up on your hardware: the correct response is to verify it and share the finding. That is how this field advances.

Danny B. Carr, Jr. Founder, Carr Digital LLC July 2026

Part I: The Threat

Chapter 1: What Quantum Computers Actually Do to Crypto

The threat isn't theoretical. It's logistical. Adversaries are vacuuming up your encrypted traffic right now to crack open later, like hoarding locked safes until someone ships the universal key. The question was never "will quantum break classical crypto?" It's "will your data still matter when it does?"

The Short Version

Most crypto books open with a wide-eyed quantum-computing explainer. This one opens with a threat model, because that's how security engineers actually think, and because "cool physics" has never once patched a production system.

The threat: A sufficiently large quantum computer running Shor's algorithm breaks RSA, ECDSA, and ECDH in polynomial time. A sufficiently large quantum computer running Grover's algorithm halves the effective key length of symmetric ciphers.

The timeline: Cryptographically relevant quantum computers do not exist today. The most optimistic credible estimates place them at least a decade away. The most pessimistic have them never arriving in useful form. Either way, nobody sane is betting their 2035 secrets on "probably never."

Why this matters now: The time to migrate is *before* the threat materializes. A secret encrypted today with X25519 may still need to be protected in 2035. NIST finalized FIPS 203, 204, and 205 in August 2024. The algorithms exist. The standards exist. OpenSSL 3.5.0 ships them natively. The only thing missing is someone actually doing the migration. (Hi. That's the job.)

Shor and Grover

Two quantum algorithms define the threat. You need them at an intuitive level, not a mathematical one, to make correct architectural decisions.

Shor's Algorithm (1994) [R1]

Peter Shor published an algorithm that solves integer factorization and discrete logarithm problems in **polynomial time** on a quantum computer.

What breaks:

- RSA (security = difficulty of factoring large integers)
- ECDSA, ECDH, X25519, X448 (security = difficulty of elliptic curve discrete log)
- Classic Diffie-Hellman (security = difficulty of integer discrete log)
- All public-key cryptography based on these problems

What Shor actually does: Most classical security problems have a periodic structure hidden in a large mathematical space. Finding this period classically requires exponential time. A quantum computer uses a quantum Fourier transform to find the period in polynomial time. Once you have the period, the factorization (or discrete log) follows directly.

Why this is total: There is no parameter size that saves RSA or ECDH from Shor. A larger key buys time against *classical* attacks. Against a sufficiently large quantum computer running Shor, a 4096-bit RSA key falls about as fast as a 2048-bit one: a small constant factor, not an exponential wall. You can't out-key the math.

Grover's Algorithm (1996)

Lov Grover published an algorithm that performs an unstructured database search in **O(√N)** time on a quantum computer, compared to **O(N)** classically.

What this means for crypto: An adversary brute-forcing a 256-bit AES key classically must try up to 2^256 keys. With Grover's algorithm on a quantum computer, they must try only 2^128 keys. The effective security is halved.

What survives:

- AES-256: effective security drops from 256-bit to 128-bit. 128-bit security is still considered unbreakable with any realistic computational budget.
- AES-128: effective security drops to 64-bit. Marginal. Migrate to AES-256.
- SHA-256: effective pre-image resistance drops from 256-bit to 128-bit. Acceptable.
- SHA-384/SHA-512: comfortable margin.

Practical guidance: Keep AES-256-GCM. Drop AES-128 for anything long-lived. The symmetric layer does not require algorithm changes, only key length considerations.

See It Run: Shor and Grover on a Simulator

Both algorithms read as abstractions until you watch them work. You can do that on a laptop, in seconds, with no quantum hardware. The lab `src/lab-quantum/quantum_threat_demo.py` runs each on qiskit's built-in statevector simulator (no `qiskit-aer`, no IBM Quantum account), and here's its captured output:

```
== Shor: factor 15 ==
  a=7  measured phase=0.25  ->  period r=4
  factors of 15: 3 x 5
  RESULT: FACTORED 15

== Grover: find 5 in a 3-qubit register ==
  iters=2 shots=2048
  most-measured: 101 = 5 (target 5, 94.5% of shots)
  RESULT: FOUND 5
```

Shor's run picks a random base coprime to 15, uses quantum period-finding to recover the period (here r = 4, from the measured phase 0.25), and the factors 3 and 5 fall out of a classical greatest-common-divisor step. That period-finding is the exact capability that ends RSA and elliptic-curve cryptography. The only thing that changes between factoring 15 and factoring an RSA-2048 modulus is scale. Grover's run finds the marked value 5 in a 3-qubit space after two iterations, landing on it ~95% of the time: the quadratic speedup that halves symmetric-key strength, made concrete.

> **Warning:** This proves the *algorithms*, not that the *machine* has arrived. Factoring 15 needs a handful of near-perfect qubits; factoring RSA-2048 needs millions of error-corrected ones, which do not exist as of this writing. The demo shows why the threat is real and total in principle, not that your keys break today. The timeline is the open question; the mathematics is not.

For an interactive version, the Quantum Computing Playground (linked in this chapter's references) runs the same two algorithms in QScript with live state-vector visualizations. The evidence for the run above is in `evidence/quantum/quantum-threat-demo.txt`; set-up is in `requirements-quantum.txt`.

The Real Threat: Harvest Now, Decrypt Later

The quantum computing timeline is uncertain. The harvest-now-decrypt-later (HNDL) threat is not.

The attack:

1. Today: adversary intercepts and records encrypted network traffic
2. Today through 2035: adversary stores the ciphertext at low cost
3. 2035+: adversary decrypts using a quantum computer once available

This attack is already happening. It's not hypothetical fan-fiction. Nation-state signals-intelligence agencies have the budget and the patience to warehouse terabytes of ciphertext indefinitely. They don't need a quantum computer today; they only need to believe one shows up before the protected information goes stale. It's recording every conversation now in a language nobody can read yet, betting the dictionary arrives before the message stops mattering. At national scale.

Which data is at risk:

- Government and military communications with long classification lives
- Medical records (HIPAA 6-year retention minimum; clinical trial data lasts decades)

- Financial transactions and account data
- Telecommunications customer records
- Intellectual property and trade secrets
- Any secret whose value persists beyond the estimated quantum timeline

Which data is NOT significantly at risk:

- TLS sessions protecting commodity web browsing (value decays rapidly)
- Short-lived session tokens
- Payment card transactions (fraud detection windows are short)

> **War story: the day the threat model got personal**
> Before security engineering I spent two decades in law enforcement, and part of that job is standing in living rooms on the worst day of a family's life. Nobody warns you about the second disaster, the quiet one that arrives a few weeks later: the surviving spouse locked out of the bank account, the insurance policy nobody can find, the email account that held everything and answers to no one. I eventually built Etergis because of those rooms: a dead-man-switch platform that keeps encrypted secrets and delivers them to your people when you no longer can.
> The quantum threat stopped being conference material the day I asked this chapter's question about my own product. An Etergis secret is long-lived by design. It sits encrypted for years, sometimes decades, and its entire value is that it still opens on that worst day. Every one of those secrets had its key wrapped with X25519, which is exactly the kind of ciphertext the harvest-now-decrypt-later adversary is warehousing. I was making a promise my cryptography couldn't keep past 2035. That realization produced the migration Chapter 9 documents, and eventually this book.
> *Lesson: the at-risk list above reads differently when something you built is on it. Run the shelf-life question against your own systems and the promises they make. That's the moment this threat stops being noise.*

Key insight if you work anywhere with long retention windows:
Telecom call detail records, health records, financial account

histories, case files, lawful-intercept archives. That's the buffet nation-states are loading their plates with, and what they have in common isn't the industry, it's the shelf life. If your organization keeps anything that'll still be sensitive in 2035, your HNDL exposure isn't a 2035 problem. It started the day you stored it.

The whole argument fits on one timeline:

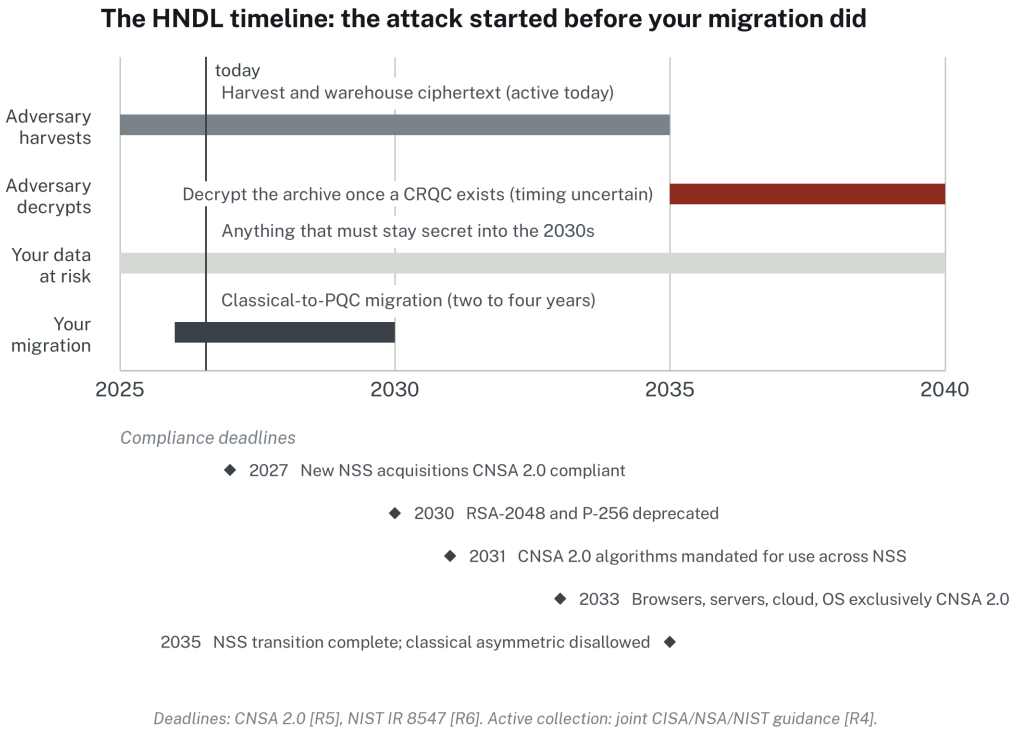

Figure 1-1. The harvest-now-decrypt-later timeline. Ciphertext recorded today sits in an adversary's warehouse while the migration deadlines close in; anything still sensitive when a cryptographically relevant quantum computer arrives is exposed retroactively, no matter what you deploy after the recording. Deadlines from CNSA 2.0 [R5] and NIST IR 8547 [R6]; active collection per the joint CISA/NSA/NIST guidance [R4]. Source: `book/figures/make_charts.py`.

What the Regulatory Bodies Say

The PQC migration is becoming a compliance requirement. Regulators have started attaching dates to it.

NSA CNSA 2.0 (September 2022) [R5]

The National Security Agency's Commercial National Security Algorithm Suite 2.0 document sets tiered migration deadlines for US government and defense contractors. Per NSA's CNSA 2.0 FAQ: software and firmware signing and traditional networking equipment must use CNSA 2.0 algorithms exclusively by 2030; web browsers, servers, cloud services, and operating systems by 2033. Per the September 2022 advisory, NSA expects the overall NSS transition to quantum-resistant algorithms to be complete by 2035 (in line with National Security Memorandum 10); 2033 is the last of the tiered category deadlines, not the stated completion date. The nearest enforcement point is January 1, 2027, when all new NSS acquisitions must be CNSA 2.0 compliant.

Specific requirements from CNSA 2.0:

- Key exchange: ML-KEM-1024
- Digital signatures: ML-DSA-87
- Hash functions: SHA-384 minimum
- Symmetric: AES-256

These deadlines are not soft. Organizations with DoD contracts or NSS data need active migration programs *now*. "We'll kick it off in 2029" is just a creative way to miss a 2030 deadline. A classical-to-PQC PKI migration typically runs two to four years once you account for vendor readiness, certificate lifecycle management, and testing.

NIST IR 8547 (Initial Public Draft, November 2024) [R6]

NIST IR 8547, *Transition to Post-Quantum Cryptography Standards*, sets the federal deprecation timeline:

- Classical algorithms at 112-bit security (RSA-2048, ECDSA P-256): **deprecated after 2030**

- All quantum-vulnerable classical asymmetric algorithms, including RSA-3072 and P-384: **disallowed after 2035**

This is the document auditors will cite in compliance reviews starting around 2027 and 2028. The time to build migration evidence is before those reviews, not during them.

CISA/NSA/NIST Quantum Readiness (2023) [R4]

In August 2023, CISA, NSA, and NIST jointly published quantum-readiness migration guidance specifically calling out HNDL: sensitive data that must stay confidential for years (government secrets, personal information, health records) is at risk from harvest-now-decrypt-later collection today, long before a quantum computer exists.

CISA categorizes critical infrastructure sectors, including telecommunications, as high-priority for PQC migration, because of the sensitivity and longevity of the data they handle.

What Replaces What

The table every migration planning session needs:

Classical Algorithm	Purpose	Quantum Vulnerability	PQC Replacement	Standard
RSA-2048/4096	Key transport, signatures	Shor (complete)	ML-KEM (key exchange), ML-DSA (signatures)	FIPS 203/204
ECDH / X25519	Key exchange	Shor (complete)	ML-KEM-768 or X25519MLKEM768 hybrid	FIPS 203
ECDSA P-256/384	Signatures	Shor (complete)	ML-DSA-65 or ML-DSA-87	FIPS 204
Ed25519	Signatures	Shor (complete)	ML-DSA-65	FIPS 204
AES-128-GCM	Symmetric encryption	Grover (partial)	AES-256-GCM	Existing FIPS 197
AES-256-GCM	Symmetric encryption	Grover (halved)	**No change needed**	Existing FIPS 197
SHA-256	Hash, HMAC, KDF	Grover (partial)	SHA-384 or SHA-512 for long-term	Existing FIPS 180-4
SHA-384/512	Hash, HMAC	Grover (acceptable)	**No change needed**	Existing FIPS 180-4

What to Migrate First

Not everything migrates at once, and pretending it does is how you torch a year and a budget. Triage with this framework:

Priority 1: Long-lived key exchange (migrate now)

Any key exchange that protects data with a long sensitivity window. This includes:

- TLS protecting data that will be stored for years
- Application-layer key wrapping (encrypted backups, secrets at rest)
- VPN tunnel keys protecting long-term archives

Action: Deploy X25519MLKEM768 hybrid in TLS configurations. This is a configuration change in nginx, OpenSSL applications, and modern TLS stacks. No code changes needed for most deployments.

Priority 2: Long-lived certificates (plan now, execute before 2028)

Root CA keys signed today will still be in certificate chains in 2035. A root CA key compromised by a quantum computer retroactively breaks all certificates it ever signed.

Action: Plan a root CA re-keying cycle using ML-DSA. This requires:

- Vendor support from your CA software (check Venafi/DigiCert/Sectigo roadmaps)
- New root distribution to trust stores
- Certificate lifecycle policy updates
- Two to three year runway minimum

Priority 3: Stored encrypted data (assess now)

Identify what encrypted data your organization retains and for how long. Anything encrypted with classical algorithms that will still be sensitive in 2035 is at HNDL risk.

Action: Inventory encrypted data stores. Re-encrypt long-lived sensitive archives with hybrid keys now. The usual suspects: subscriber and customer records, clinical and claims data, financial account histories, case files, and anything an archivist would call permanent.

Priority 4: Short-lived TLS (monitor, defer)

TLS sessions protecting commodity web traffic with no long-term value are low priority. The data decrypted in 2035 was stale in 2025. Focus budget on the categories above.

Where This Leaves You

You have the threat model: harvest-now-decrypt-later, Shor, Grover, and a priority framework for what moves first. The threat is credible and already active in its HNDL form. The response, NIST FIPS 203, 204, and 205, is finalized and implementable today on stock OpenSSL 3.5.5. If you hold long-lived sensitive data, government contracts, or critical infrastructure obligations, the migration is a question of *when* and *in what order*, not *whether*.

The next chapter answers the question this framework quietly assumes you can already answer: why trust ML-KEM, ML-DSA, and SLH-DSA at all? That answer is a seven-year, sixty-nine-submission story with a body count. A Round 3 finalist fell to a laptop over a weekend. An alternate fell in under an hour, one week after NIST announced the winners. Neither needed a quantum computer. It matters for the same reason the threat model does: you're about to bet production systems on it.

References

- [R1] Shor, P.W.: Polynomial-Time Algorithms for Prime Factorization: https://doi.org/10.1137/S0097539795293172
- [R4] CISA/NSA/NIST: Quantum-Readiness, Migration to Post-Quantum Cryptography (Aug 2023): https://www.cisa.gov/resources-tools/resources/quantum-readiness-migration-post-quantum-cryptography
- [R5] NSA: "Announcing the Commercial National Security Algorithm Suite 2.0," Cybersecurity Advisory (PP-22-1338), September 2022. Full text: https://post-quantum-measured.pages.dev/CSA_CNSA_2.0_ALGORITHMS.pdf
- NSA, CNSA 2.0 FAQ, v2.1 (December 2024; U/OO/194427-22 | PP-24-4014): the FAQ that resolves the provisional-algorithm-name ambiguity in the 2022 advisory above: https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF
- [R6] NIST IR 8547: Transition to Post-Quantum Cryptography Standards (IPD): https://csrc.nist.gov/pubs/ir/8547/ipd
- Mosca, M.: Cybersecurity in an Era with Quantum Computers: Will We Be Ready? (2018 IEEE S&P): https://doi.org/10.1109/MSP.2018.3761723

Further reading: hands-on and research

For readers who want to *see* quantum computing rather than only read about it, and to follow the primitive-level research this book builds on (pointers recommended in J.-P. Aumasson, *Serious Cryptography*, 2nd ed., No Starch Press, 2024, Ch. 14):

- IBM Quantum: cloud quantum computing platform with visual circuit composer: https://quantum.ibm.com
- Quantum Computing Playground: browser-based quantum simulator with visualizations: https://www.quantumplayground.net (third-party; availability varies)
- PQCrypto.org: the post-quantum cryptography research community hub, and the associated **PQCrypto** conference series: https://pqcrypto.org