The Digital Social Contract: A Lawfare Paper Series

NIST's Software Un-Standards

Bryan H. Choi[*]

March 2024

NIST's standardization efforts in computing technologies retreat from traditional engineering standards in favor of permissive, open-ended frameworks, undercutting the hope that NIST standards can establish a legal standard of reasonable care for software safety.

Over the last decade, the White House has recast the National Institute of Standards and Technology (NIST) in a new leading role of setting national cyber standards. Beginning in 2013, the Obama, Trump, and Biden administrations issued a series of executive orders directing NIST to develop a "cybersecurity framework" and to define specific cybersecurity "performance goals" for critical infrastructure.[1] The Trump and Biden administrations have extended NIST's purview to artificial intelligence (AI), giving

[*] Associate Professor of Law and Computer Science & Engineering, The Ohio State University. I thank Derek Bambauer, Bridget Dooling, Brett Frischmann, Gus Hurwitz, Asaf Lubin, Nick Nugent, Blake Reid, Alan Rozenshtein, Melanie Teplinsky, David Thaw, Rebecca Wexler, and members of the Law and Technology Workshop Series for helpful input at early stages of this project. I also thank Rebecca Fordon, Michael Adamo, Kevin Gibbons, and Aditya Medicherla for excellent research assistance. This work was supported in part by NSF CCF-2131531.

[1] See "Improving Critical Infrastructure Cybersecurity," Executive Order 13636, 78 Fed. Reg. 11739, Feb. 19, 2013 (directing NIST "to lead the development of a framework to reduce cyber risks to critical infrastructure," which includes "a set of standards, methodologies, procedures, and processes ... to address cyber risks"); "Promoting Private Sector Cybersecurity Information Sharing," Executive Order 13691, 80 Fed. Reg. 9347, Feb. 20, 2015; "Commission on Enhancing National Cybersecurity," Executive Order 13718, 81 Fed. Reg. 7441, Feb. 12, 2016; "Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure," Executive Order 13800, 82 Fed. Reg. 22391, May 16, 2017; "Improving the Nation's Cybersecurity," Executive Order 14028, 86 Fed. Reg. 26633, May 17, 2021; see also "National Security Memorandum on Improving Cybersecurity for Critical Infrastructure Control Systems," July 28, 2021, https://perma.cc/828S-HZVL (ordering NIST and other agencies to "develop and issue cybersecurity performance goals for critical infrastructure to further a common understanding of ... baseline security practices").

NIST the task of developing guidelines, standards, and best practices to make AI systems more "safe, secure, and trustworthy."[2] Congress has bolstered those executive orders with legislative actions.[3]

That delegation to NIST has renewed hopes for a quick-fix solution to the problem of software liability. Commentators have begun to assert that NIST standards could provide a simple compliance mechanism that would satisfy a "reasonable" duty of care.[4] For example, in 2018, when the Ohio legislature addressed the issue of data breach lawsuits, it leaned primarily on NIST standards to establish a safe harbor.[5] The resulting Data Protection Act shields Ohio businesses from liability for data breaches as long as those businesses implement appropriate cybersecurity controls such as NIST's Cybersecurity

[2] See "Maintaining American Leadership in Artificial Intelligence," Executive Order 13859, 84 Fed. Reg. 3967, Feb. 11, 2019 (directing NIST to develop "technical standards and related tools in support of reliable, robust, and trustworthy systems that use AI technologies"); "Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence," Executive Order 14110 § 4, 88 Fed. Reg. 75191, 75196, Oct. 30, 2023 (directing NIST to "[e]stablish guidelines and best practices, with the aim of promoting consensus industry standards, for developing and deploying safe, secure, and trustworthy AI systems").

[3] See Pub. L. No. 116-283, 134 Stat. 4523 (2021) (directing NIST to develop "technical standards and guidelines that promote trustworthy artificial intelligence systems"); Internet of Things Cybersecurity Improvement Act of 2020, Pub. L. No. 116-207, 134 Stat. 1001 (2020); NIST Small Business Cybersecurity Act, Pub. L. No. 115-236, 132 Stat. 2444 (2018).

[4] See William McGeveran, "The Duty of Data Security," *Minnesota Law Review* 103 (2019): 1135, 1164 (stating that the "peaceful coexistence" of the NIST Cybersecurity Framework and other independent industry standards "underscores the broad consensus among security experts about the core elements of the duty of data security"); Scott J. Shackleford et al., "Defining 'Reasonable' Cybersecurity: Lessons From the States," *Yale Journal of Law & Technology* 25 (2023): 86, 120 (finding that, in defining "reasonable" cybersecurity, state laws seem to be converging on some combination of the NIST Cybersecurity Framework and the Center for Internet Security (CIS) Top 20 Security Controls); see also id. at 139 (finding that the NIST Cybersecurity Framework is "the dominant cybersecurity framework used by most small and medium-sized businesses"). But see Charlotte A. Tschider, "Locking Down 'Reasonable' Cybersecurity Duty," *Yale Law & Policy Review* 41 (2023): 75, 111 (observing that the flexible nature of cybersecurity standards such as the NIST Cybersecurity Framework creates "great difficulty" for courts to determine "whether an organization actually employed reasonable security practices").

[5] See Data Protection Act, Ohio S.B. 220 (2018), codified at Ohio Rev. Code ch. 1354, https://codes.ohio.gov/ohio-revised-code/chapter-1354. The statute offers a safe harbor to businesses that create, maintain, and comply with a written cybersecurity program that "reasonably conforms to an industry recognized cybersecurity framework" such as NIST's Cybersecurity Framework.

Framework.[6] The Ohio law has become a centerpiece in the argument that NIST standards could establish a legal baseline for acceptable software practices.

Such invocations of NIST's cyber frameworks as a standard of care raise the question whether they are adequate. The prevailing view is that NIST is a trustworthy, nonpartisan body that promulgates reliable, scientific standards. That trust in NIST reflects the sound reputation NIST has earned in establishing uniform standards across a broad range of disciplines. For those who believe federal agencies should assume a more prominent role in cyber governance, NIST appears to be a natural vessel for that agenda.

Yet, the turn to NIST calls into question whether such faith is justified. Typically, commentators invoke NIST as a black-box solution and mention NIST's competence only in passing.[7] Little is said about the content of NIST's software standards, or whether compliance with those standards will meaningfully improve the safety and quality of software systems. Nor is there much discussion of NIST's role as an institutional body. To be sure, NIST enjoys a sterling reputation in traditional areas of metrics and standardization such as the physical sciences. But as I have argued elsewhere, software is different in ways that often defy measurement.[8]

---

[6] See id. at ch. 1354.03(A)(1) (listing NIST standards as the first three of six "industry recognized" cybersecurity frameworks). Certain regulated entities can also qualify via compliance with their governing statutes. See id. at ch. 1354.03(B)(1). See generally David J. Oberly, "Ohio's Data Protection Act," *Ohio Lawyer*, July 1, 2019 (noting that Ohio's Data Protection Act is "the first law in the country to provide incentives to businesses to implement certain cybersecurity controls through the utilization of an affirmative defense to liability in the wake of a data breach"); Dennis Hirsch et al., "Promoting Better Cybersecurity: An Analysis of the Ohio Data Protection Act," March 25, 2019, at 8.

[7] See, e.g., David Thaw, "The Efficacy of Cybersecurity Regulation," *Georgia State University Law Review* 30 (2013): 287, 369 ("[R]eferencing current standards on encryption, such as those promulgated by NIST, provides an excellent, flexible, and adaptive solution. Developing standards is among NIST's core competencies, and it publishes Federal Information Processing Standards on a wide variety of topics, including encryption."); Justin (Gus) Hurwitz, "Cyberensuring Security," *Connecticut Law Review* 49 (2017): 1495, 1504–5 (describing the NIST Cybersecurity Framework as the "gold-standard"); Charlotte A. Tschider, "Medical Device Artificial Intelligence: The New Tort Frontier," *BYU Law Review* 46 (2021): 1551, 1594 n.204 (claiming that NIST is "the agency best positioned to regulate AI technologies or promote standardization").

[8] See Bryan H. Choi, "Software as a Profession," *Harvard Journal of Law & Technology* 33 (2020): 557 (explaining the expert consensus that software's "essential complexity" exceeds the capacity of conventional engineering methods and defies standardization efforts); Bryan H. Choi, "Institutional Choice in Software Safety Standards," *Hastings Law Journal* 73 (2022): 1461 (arguing that the centralized agency model lacks a comparative advantage when the central obstacle is scientific indeterminability due to software complexity); accord Thaw, supra note 7, at 302 ("Professionals and regulators evaluate information security outcomes as a function of whether certain practices are followed, not whether those practices are effective. This approach

The first part of this paper describes NIST's origin and involvement in the development of early computing and data processing standards. It then unpacks NIST's new frameworks for cybersecurity, secure software development, and artificial intelligence. Although the agency played a critical role until the late 1970s, its influence waned rapidly beginning in the late 1980s, and it was virtually invisible by the 1990s. Several high-profile cybersecurity incidents have now thrust NIST back into action. After the hiatus, NIST has touted voluntary "risk management frameworks" in lieu of formal technical standards.

The second part argues that NIST's software standards are not "standards" in the conventional sense of bringing uniformity to a practice. Instead, NIST has gravitated toward self-governance frameworks that tolerate a broad range of software practices. Accordingly, NIST's experience offers two lessons.

First, there may not exist a single, consistent "reasonable" standard of care for software liability. Accordingly, lawmakers looking to craft a software standard of care would be wise to embrace hybrid elements of self-governance and nonstandard software practices. Paradoxically, efforts to incorporate NIST's software "standards" directly into the software developer's duty of care tacitly ratify this approach, despite seeming to do the opposite.

Second, the central agency model is unlikely to offer easy shortcuts for determining when software liability does or does not attach. Because NIST is the standard-bearer for federal standard-setting, NIST's retreat from software standardization is the strongest possible indictment against centralized, uniform mandates. If NIST's expertise is trustworthy, then policymakers should heed the signal that alternate mechanisms are needed.

A BRIEF HISTORY OF NIST

NIST is a nonregulatory federal agency situated within the U.S. Department of Commerce. NIST's official mission is to provide measurements, calibrations, and quality assurance techniques to promote "commerce, technological progress, improved product reliability and manufacturing processes, and public safety."[9] Historically, that mission has focused primarily on economic concerns such as international trade, scientific innovation, federal procurement, and budgetary waste, although there has been substantial overlap with noneconomic concerns such as national security during times of war.

is, in part, due to an inability to measure the efficacy of such practices because demonstrating success is often an exercise in 'proving a negative.'").

[9] Technology Competitiveness Act, Pub. L. No. 100-418 § 5101, 102 Stat. 1426 (1988), codified at 15 U.S.C. § 271(b)(1) (amending Pub. L. No. 56-177, 31 Stat. 1449 (1901)).

The entity now known as NIST was established in 1901 as the National Bureau of Standards (NBS).[10] Although the U.S. Constitution authorizes Congress to "fix the Standard of Weights and Measures,"[11] prior efforts had been "puny" and ineffective, resulting in "a whole galaxy of entirely arbitrary standards affecting almost every measurable quantity."[12] In establishing NIST, Congress sought to keep pace with scientific progress and to be more competitive on the international stage.[13] Accordingly, Congress delegated broad authority to the new agency to manage "custody of the standards," including the power to construct new standards.[14]

NIST's involvement in computing began in 1946, when it fielded requests from other agencies to assist in the procurement, development, and maintenance of computers for federal government use.[15] Early efforts focused primarily on provision of basic computer services and ad hoc advisory support to other federal agencies.[16] Building on this expertise, NIST engaged in a broad range of research efforts to

[10] Pub. L. No. 56-177, 31 Stat. 1449 (1901); see also Rexmond C. Cochrane, U.S. Department of Commerce, *Measures for Progress: A History of the National Bureau of Standards*, 2nd ed. (1974), 47 (describing the contentious history leading up to the creation of NBS); James F. Schooley, *Responding to National Needs* (2000), 7–11. Congress renamed the agency from NBS to NIST in 1988. See 15 U.S.C. § 271(b)(1). For consistency, the remainder of this paper will use "NIST" to refer to the agency both before and after the name change.

[11] U.S. Constitution, Article I, § 8, cl. 5.

[12] Cochrane, supra note 10, at 54.

[13] See id. at 38–39 (explaining the country's new stature as a world power following the Spanish-American War); see also 56 H. Rep. 1452 (1900) (House committee report recommending the creation of a national standardizing bureau); 56 S. Doc. 70 (1900) (letter from Secretary of the Treasury Lyman Gage observing that necessary scientific instruments of precision are "too frequently procured from abroad, owing to our own lack of facilities for standardizing" and that it is "absolutely essential that American manufacturers of such apparatus have access to a standardizing bureau equivalent to that provided for the manufacturers of other countries, notably Germany and England").

[14] Pub. L. No. 56-177 § 2.

[15] See U.S. Department of Commerce, "1969 Technical Highlights of the National Bureau of Standards" (1970), at 7 (describing early requests from the Bureau of the Census, Office of Naval Research, and Office of the Chief of Ordnance, Department of the Army); see also Elio Passaglia, *A Unique Institution* (1999), 41; John L. McClellan, Chairman, Committee on Government Operations, U.S. Senate, "Report to the President on the Management of Automatic Data Processing" ("Gordon Report") (1965), 55.

[16] NIST built several of its own computers. See U.S. Department of Commerce, supra note 15, at 7–10. For example, projects during the 1950s included sorting and file merging for the Public Housing Administration, optical character recognition for the Social Security Administration, and automatic mail sorting for the Post

improve computer hardware components and computational techniques.[17] NIST personnel also participated in external initiatives to develop higher-level programming languages such as ALGOL and COBOL.[18]

The Brooks Act of 1965 marked a sea change, reorganizing those computer-related activities under a new subdivision called the Center for Computer Sciences and Technology (CCST).[19] Much of the thrust was on streamlining federal use of computing resources. But the most salient change was that the Brooks Act instructed NIST to develop uniform federal standards for "automatic data processing" (ADP) equipment.[20]

To fulfill those duties, NIST established a new framework of Federal Information Processing Standards (FIPS). In theory, the FIPS framework was broad enough to have supported an expansive role in setting standards across the software and computing industry. In practice, however, the FIPS project proved to be less influential than hoped. Today, nearly all FIPS have been withdrawn.

Early Computing Standards

NIST devoted substantial effort to the FIPS framework, but progress was slow and constrained by resource limitations. For example, in its five-year report, NIST stated that available resources "forced it

Office Department. NIST also shared its own computers with other agencies that "either did not have computers or were not fully equipped in this area." Id. at 13–14.

[17] Id. at 11.

[18] Id. at 13.

[19] Brooks Act, Pub. L. No. 89-306, 79 Stat. 1127 (1965). The CCST was renamed in 1972 the Institute for Computer Sciences and Technology (ICST); in 1988 the National Computer Systems Laboratory (NCSL); in 1991 the Computer Systems Laboratory (CSL); and in 1996 the Information Technology Laboratory (ITL). See NIST, "ITL History Timeline 1950–Present," https://perma.cc/66Q5-EQZS. Today, NIST operates six laboratory programs, including communications technology, engineering, information technology, material measurement, neutron research, and physical measurement. See NIST, "NIST Organization Structure," https://perma.cc/P7QZ-RQBX.

[20] The Brooks Act instructed NIST to (1) provide scientific and technological advisory services to other agencies with regard to ADP equipment; (2) recommend uniform federal ADP standards; and (3) undertake research in computer science and technology as needed to fulfill those responsibilities. See Pub. L. No. 89-306, 79 Stat. at 1128; see also Passaglia, supra note 15, at 500–4. The development of such standards also related to cost-efficiency concerns, by allowing the federal government to reduce reliance on "single vendor" procurement practices. See NBS, "Brooks Bill Issue Study of the National Bureau of Standards," (1971), VI.6–VI.7 [hereinafter "NBS Five-Year Report"].

to focus on exceedingly modest, short-term goals."[21] And in its ten-year report, NIST noted that the General Accounting Office had issued a dozen reports stating that NIST was unable to fulfill its responsibilities "due to lack of financial and manpower resources."[22]

By the end of the first decade, NIST had published only twenty-seven FIPS, despite devoting nearly four-fifths of its budget to the task.[23] Moreover, most of these early publications addressed only simple, low-level issues such as data storage media (e.g., magnetic tape, punch cards) or regional geographic codes (e.g., states, counties, congressional districts).[24] NIST estimated that development of new standards took an average of three years if done internally, or five years if working with an external industry standards group such as the American National Standards Institute (ANSI).[25]

During the second decade, from 1976 through 1986, NIST accelerated its efforts and issued ninety-seven new FIPS, in areas ranging from encryption and computer security, to programming languages, data elements, interfaces, and storage media.[26] Those topics hewed closely to the "priorities" identified in 1969—more than fifteen years earlier—evincing the ponderous pace of NIST's standardization work.[27] To offset the slow pace of the FIPS process, NIST also launched a Special Publication (SP-500) series

[21] See "NBS Five-Year Report," supra note 20, at VI.3.

[22] See Grace Burns & Shirley Radack, U.S. Department of Commerce, "A Ten Year History of National Bureau of Standards Activities Under the Brooks Act" (1977), at 5 [hereinafter "NBS Ten-Year Report"].

[23] See id. at 1 ("The preponderance of NBS effort over the last ten years (i.e., nearly four-fifths of its directly appropriated funds) has been directed to the development of ADP standards and guidelines.").

[24] See Shirley M. Radack, "The National Computer Systems Laboratory: An Overview of Technical Activities," *Computer Standards & Interfaces* 10 (1990): 191, 192 ("Some of the early FIPS included: [ASCII]; perforated tape, punch cards, and recorded magnetic tape standards; standard data elements and codes for representing geopolitical entities, map coordinates, time and measurement units; COBOL programming language standard; standards for the sequencing of data for transmission over telephone lines."). This narrow focus was reflected by the initial task groups that NIST established in 1969: (1) objectives and requirements for standards; (2) data terminals and data interchange systems requirements; (3) character subsets, sign conventions, and packing techniques; (4) subsections on standards for use in requests for proposals; (5) vocabulary; (6) computer magnetic tape; (7) magnetic tape labels; (8) format description for information interchange; (9) COBOL; and (10) computer systems performance evaluation. See "NBS Five-Year Report," supra note 20, at VI.19–VI.21.

[25] See id. at 33–34.

[26] See NBS, "FY 1986 Annual Report" (1986), at 63–68.

[27] See "NBS Five-Year Report," supra note 20, at VI.11–VI.18.

on computer systems technology, which allowed NIST to provide informal guidance on topics of interest. Between 1977 and 1986, NIST issued 144 publications in the SP-500 series.[28]

This burst of activity proved to be the high-water mark. Beginning in the late 1980s, Congress shifted NIST's role and then dramatically pared back the agency's role in software standardization activities.[29] Although NIST retains authorization to publish new FIPS, it has issued only seven since 1995 and none since 2015.[30] Likewise, NIST released another ninety-three publications in the SP-500 technical series up through 1996, after which activity ceased almost entirely. As a result, NIST was essentially invisible through the most defining years of the internet era.

Three categories of FIPS are illustrative of the structural challenges of software standardization. First, the programming language standards demonstrate NIST's inability to keep pace with the speed of private-sector innovation. Second, NIST's fledgling attempts to govern the overall software development lifecycle process reveal that the value of FIPS standardization was inversely proportional to the complexity of the task. Third, NIST's relative success with data encryption standards—which are the main FIPS still actively maintained by the agency[31]—is the exception that proves the rule, by showing that federal software standards work best when the task is narrowly scoped.

[28] See NIST, NIST Technical Series Publication List: SP500, https://perma.cc/82FL-PHZ8.

[29] Three key congressional actions during this period were the Computer Security Act of 1987, Pub. L. No. 100-235, 101 Stat. 1724 (1988) (directing NIST to develop standards and guidelines on security and privacy of digital information); the Omnibus Trade and Competitiveness Act, Pub. L. No. 100-418 § 5101, 102 Stat. 1107, 1426 (1988) (renaming NIST and directing it to assist industry in facilitating more rapid commercialization of new scientific discoveries); and the National Technology Transfer and Advancement Act of 1995, Pub. L. No. 104-113, 110 Stat. 775 (1996) (directing NIST to eliminate unnecessary duplication of private-sector technical standards activities, and directing all federal agencies to use technical standards developed by voluntary consensus standards bodies). See generally Schooley, supra note 10, at 613 (explaining that the name change from NBS to NIST in 1988 "was rooted in the growing awareness in Congress that the American enterprise was faltering in international competition" and that Congress added "substantial new responsibilities to the NBS mission"); id. at 640 (citing a 1988 congressional committee report that "called attention to the potential damage to NIST programs from consistent underfunding of the agency" and warning that "NIST's ability to preserve its scientific competence might suffer from an overemphasis on technology transfer").

[30] See NIST, NIST Technical Series Publication List: FIPS, https://perma.cc/UVK7-58UY (listing FIPS 196 (1997) through FIPS 202 (2015)). NIST recently published three draft FIPS on post-quantum cryptography. See NIST, "Comments Requested on Three Draft FIPS for Post-Quantum Cryptography," Aug. 24, 2023, https://perma.cc/YNJ9-4938.

[31] Of the nine FIPS still currently maintained, six relate to cryptographic standards, one involves identity verification, and the remaining two deal with information security. See NIST, Current Approved and Draft

Programming Languages

Standardization of high-level programming languages was an early priority area for NIST. The central motivation was to facilitate portability of programs across different types of computer systems, thus saving on costs in ongoing software development and maintenance activities.[32] A secondary goal was to facilitate the production of error-free code.[33]

In 1972, NIST adopted COBOL as the first federal standard programming language.[34] NIST had been intimately involved in the development of COBOL since 1959, so it was logical that NIST would continue to support and promote its use.[35] A particularly acute problem at the time was that COBOL compilers were inconsistent across different vendors.[36] Compilers are low-level tools that translate human-written software code to machine-readable instructions. NIST launched a new service that validated whether COBOL compilers were properly implemented on federal computers in accordance

FIPS, https://csrc.nist.gov/publications/fips (listing FIPS 140, 180, 186, 197, 198, 199, 200, 201, and 202); NIST, Withdrawn FIPS Listed by Number, https://perma.cc/VV8V-BDBK.

[32] See NBS, FIPS PUB 23, "Objectives and Requirements of the Federal Information Processing Standards Program" (1973), at 4; John V. Cugini, NBS, Special Publication 500-117, "Vol. 1: Selection and Use of General-Purpose Programming Languages—Overview," (1984), at iii (stating that "good language standards make it easier and less costly to transport software from one language processor to another"); John V. Cugini et al., NBS, Special Publication 500-70/1, "NBS Minimal BASIC Test Programs—Version 2, User's Manual" (1980), at 9 ("At bottom, however, there is one result essential to the success of a [programming language] standard: program portability. The same program should not evoke perniciously different behavior in different implementations.").

[33] See Karl N. Levitt et al., NBS, Special Publication 500-67, "The SRI Hierarchical Development Methodology (HDM) and Its Application to the Development of Secure Software" (1980), at 4 (acknowledging "new features [that] have been incorporated [in programming languages] to aid the programmer in producing more error-free programs," but "reject[ing] the view that programming languages should continue to become more complex in order to provide those features").

[34] See NBS, FIPS PUB 21, "COBOL" (1972). The American National Standards Institute (ANSI) issued an official standard in 1968. It took four more years for NIST to issue a FIPS incorporating wholesale the ANSI standard.

[35] See "NBS Ten-Year Report," supra note 22, at 61 (noting that although "[n]either the Federal Government nor NBS control COBOL development ... it was largely NBS that provided technical guidance, financial support for publications, and—most important, perhaps—a continuing resolve to see to completion the first attempt at a common, business-oriented language standard").

[36] Id. at 62; James H. Burrows, "Information Technology Standards in a Changing World: The Role of the Users," *Computer Standards & Interfaces* 15 (1993): 49, 53.

with the FIPS.[37] NIST also provided guidance and seminars on how to program in COBOL.[38] This validation service was an early success. As a result of NIST's work, COBOL became the leading programming language used within the federal government during the 1970s.[39]

But NIST was slow to keep up as the pace of software innovation accelerated, perhaps because it believed COBOL would outcompete other nonstandard languages.[40] After NIST completed its review of COBOL in 1972, the next FIPS for FORTRAN and BASIC lagged until 1980.[41] Additional programming language FIPS were released in 1985 (Pascal, Ada), 1986 (MUMPS), 1987 (SQL), and 1991 (C),[42] after which NIST abandoned the enterprise. Most of the standardization work was performed by ANSI, a nongovernmental organization, of which NIST was a participating member. Each FIPS simply incorporated ANSI's specification by reference, and mandated that all government software must conform with the ANSI standard. Typically, the ANSI process took years to complete, with final

[37] See Burrows, supra note 36, at 53 ("As a result, the U.S. Government launched a project to develop validation systems and to require validation of COBOL compilers acquired by agencies."); NBS, FIPS PUB 80, "Guide for the Implementation of Federal Information Processing Standards (FIPS) in the Acquisition and Design of Computer Products and Services," (1980), at 59–61.

[38] See "NBS Ten-Year Report," supra note 22, at 61 (listing NIST activities relating to COBOL).

[39] See id. at 61 (stating that more than 61 percent of domestic federal installations use COBOL); see also Martha Mulford Gray, NBS, Special Publication 500-79, "An Assessment and Forecast of ADP in the Federal Government" (1981), at ix (estimating that more than 50 percent of federal installations were using COBOL as their principal programming language).

[40] See, e.g., Gray, supra note 39, at 2-3, 2-8 (wrongly predicting federal use of COBOL to increase, and use of newer languages such as BASIC to decrease, because "[e]ven if more sophisticated and user-friendly software is developed during the next 5 years, vendors will face a difficult marketing task in educating users, programmers, and systems managers to new technologies").

[41] See NBS, FIPS PUB 68, "Minimal BASIC" (1980) (adopting ANSI standard); NBS, FIPS PUB 69, "FORTRAN" (1980) (same). FORTRAN was developed in the late 1950s, while the first version of BASIC was created in 1963. NBS also approved updated versions of the COBOL standard in 1975, 1990, and 1995. See NIST, FIPS PUB 21-4, "COBOL" (1995) (superseding FIPS 21-3 and FIPS 21-2).

[42] See NBS, FIPS PUB 109, "PASCAL" (1985) (adopting ANSI standard); NBS, FIPS PUB 119, "Ada" (1985) (same); NBS, FIPS PUB 125, "MUMPS (Massachusetts General Hospital Utility Multi-Programming System)" (1986) (same); NBS, FIPS PUB 127, "Database Language SQL" (1987) (same); NBS, FIPS PUB 160, "C" (1991) (same).

approval of the FIPS taking another couple of years. NIST continued to introduce new compiler validation services for each approved language,[43] but the development of tests was costly and slow.[44]

This stagnation undermined NIST's efforts at standardization. For example, NIST took a dim view of nonstandard language elements.[45] But software developers overwhelmingly resisted the crippling restrictions, eventually forcing NIST to admit that such nonstandard elements "can be very useful."[46] Likewise, NIST initially mandated that government software use should be limited to approved programming languages only, believing that the government's purchasing power would encourage greater standardization across the industry.[47] But NIST removed that language by the mid-1980s, as major commercial entities such as Microsoft and Apple opted instead to use more capable, unapproved

[43] See "Computer Systems Laboratory—An Overview," *Computer Standards & Interfaces* 14 (1992): 445, 450–51 (stating that "[t]esting programming language compilers for conformance to FIPS programming language standards ... continued to be an important service," and that NIST "continued to publish quarterly the *Validated Products List* which is a collection of registers listing implementations that have been validated for conformance to FIPS"). But see W.S. Brainerd, "The Programming Language Standards Scene, Ten Years On: Fortran," *Computer Standards & Interfaces* 16 (1994): 459, 463 (noting that as of 1993, NIST "indicated no interest" in developing new validation suites).

[44] See Burrows, supra note 36, at 54 ("Tests are sometimes developed as part of the standards process, but more often are developed later, and are costly to produce.").

[45] See NBS, FIPS PUB 21-1, "COBOL" (1975), at 4 ("Programs should, to the extent practicable, be limited to the elements of one of the specified levels of Federal Standard COBOL. It should be recognized that the use of any non-standard language elements may compromise interchangeability of programs between various systems or may complicate future conversion to a replacement system. Extensions should, therefore, be employed only when their use will result in efficiencies that clearly outweigh the difficulties they may cause").

[46] FIPS PUB 109, supra note 42, at 2 ("Although non-standard language features can be very useful, it should be recognized that their use may make the interchange of programs and future conversion to an extended Pascal standard or replacement processor more difficult and costly."); FIPS 127, supra note 42, at 2 (same); FIPS 160, supra note 42, at 2 (same). But see FIPS 119, supra note 42, at 2 ("The standard for Ada adopted herein ... does not allow conforming implementations to extend the language.").

[47] At that time, the only approved languages were COBOL, BASIC, and FORTRAN. See FIPS 68, supra note 41, at 2 ("Federal standards for high level programming languages shall be used for computer applications and programs that are developed or acquired for government use. ... The use of specific programming languages is limited to the approved Federal Information Processing Standards languages."); FIPS 69, supra note 41, at 2 (same).

languages.[48] Because the FIPS became more obstructive than useful, they were often ignored. Although NIST tried to institute a strict process for obtaining waivers from FIPS requirements,[49] it ultimately conceded that informal, unwritten waivers had become a necessary practice.[50]

The ponderous pace of FIPS approvals meant that government software developers were expected to use older, clumsier programming languages, even as the commercial sector forged ahead with newer, nimbler languages.[51] NIST had predicted that federal standard programming languages would win out because they would reduce development and maintenance costs over the long term.[52] Instead, counterintuitively, the rigidity of the FIPS program pushed the government to shift from in-house development to commercial procurement, as off-the-shelf software provided substantially more features at substantially lower cost.[53] It also subverted the authority of NIST and the FIPS program, as noncompliance became the norm across the software community.

---

[48] See, e.g., Derek Jones, "The Programming Language Standards Scene, Ten Years On: C," *Computer Standards & Interfaces* 16 (1994): 495, 496 (attributing the popularity of the nonstandard language C to its "traditional spirit" of empowering software developers: "Trust the programmer. Don't prevent the programmer from doing what needs to be done. ... Make it fast, even if it is not guaranteed to be portable."); Richard M. De Morgan, "The Programming Language Standards Scene, Ten Years On: C++," *Computer Standards & Interfaces* 16 (1994): 531, 534 (attributing the mass appeal of C++ to its support of more sophisticated features and to its availability at affordable prices).

[49] See FIPS 21-1, supra note 45, at 4–5 (allowing waivers only by "[h]eads of agencies" and requiring waivers to be "obtained before ... implementation or acquisition"); FIPS 68, supra note 41, at 4 (requiring written requests for waiver and that "[n]o agency shall take any action to deviate from the standard prior to the receipt of a waiver approval from the Secretary of Commerce"); FIPS 69, supra note 41, at 4 (same); see also Computer Security Act of 1987, Pub. L. No 100-235 § 4, 101 Stat. 1724, 1728 (1988).

[50] See FIPS 127, supra note 42, at 4 (allowing "agency heads" to approve requests for waiver, not just the secretary of commerce); FIPS 160, supra note 42, at 4 (expanding the authority of agency heads to "also act without a written waiver request when they determine that conditions for meeting the standard cannot be met").

[51] See Cugini, NBS SP 500-117, supra note 32, at 2, 35–39 (1984) (noting extensive government use of software programs written in unapproved languages such as C or noncompliant versions of BASIC).

[52] See Helen M. Wood, "Emerging Software Standards: Opportunity and Challenge," *Computer Standards & Interfaces* 6 (1987): 239, 242 (predicting that the "prospect of achieving significant productivity increases and reduced costs" would "lur[e] the Government and other users" to embrace federal programming language standards).

[53] Compare Wilma M. Osborne, NBS, Special Publication 500-130, "Executive Guide to Software Maintenance," at 7 (1985) ("The Federal Government continues to custom develop more than 90% of its software."), with Defense Science Board, U.S. Department of Defense, "Report of the Defense Science

Software Development Lifecycle

More ambitiously, NIST hoped to standardize the entire software development process. As early as 1971, NIST turned its attention to "developing a technical basis for the documentation, validation, correctness, quality control during development, and sharing of software."[54] In 1973, NIST sponsored a planning workshop for a "Software Engineering Handbook."[55] By 1983, NIST reported that poor software development practices were generating enormous wasteful costs for the government.[56] Those initial explorations led to an outpouring of FIPS and informal guidance on topics including

Board Task Force on Acquiring Defense Software Commercially" (1994), at C-1 [hereinafter "Report on Acquiring Defense Software Commercially"] (concluding that commercial software development processes are "more flexible and open" and thus are able "to [field] a system sooner and evolve it to include more capability at significant cost savings"). Cf. Robert W. Hahn & Anne Layne-Farrar, "The Law and Economics of Software Security," *Harvard Journal of Law & Public Policy* 30 (2006): 283, 347 (noting that because "security costs money and reduces features," most government agencies "pay little or no attention to [software] security issues" with over half of the agencies receiving a D or F in an annual review completed in 2003).

[54] See "NBS Ten-Year Report," supra note 22, at 32 ("The Institute has been working toward issuance of a 'Software Engineering Handbook' which will provide to Federal systems planners the most complete compilation of meaningful software design and performance measurement practices developed through this technical activity.").

[55] See Selden L. Stewart, NBS, Technical Note 832: "Report on Planning Session on Software Engineering Handbook" (1974).

[56] See Roger J. Martin & Wilma M. Osborne, NBS, Special Publication 500-106, "Guidance on Software Maintenance" (1983), at 2 (estimating that "60% to 70% of the total application software resources are spent on software maintenance.").

documentation;[57] software planning and design;[58] software development tools;[59] validation, verification, and testing;[60] and maintenance.[61]

---

[57] See NBS, FIPS PUB 38, "Guidelines for Documentation of Computer Programs and Automated Data Systems" (1976); NBS, FIPS PUB 105, "Guideline for Software Documentation Management" (1984); see also Mitchell A. Krasny, NBS, Special Publication 500-15, "Documentation of Computer Programs and Automated Data Systems" (1977); Albrecht J. Neumann, NBS, Special Publication 500-87, "Management Guide for Software Documentation" (1982); A.J. Neumann, NBS, Special Publication 500-94, "NBS FIPS Software Documentation" (1982) (workshop proceedings).

[58] See NBS, FIPS PUB 64, "Guidelines for Documentation of Computer Programs and Automated Data Systems for the Initiation Phase" (1979); see also Dennis W. Fife, NBS, Special Publication 500-11, "Computer Software Management: A Primer for Project Management and Quality Control" (1977); Karl N. Levitt et al., NBS, Special Publication 500-67, "The SRI Hierarchical Development Methodology (HDM) and Its Application to the Development of Secure Software" (1980); Dolores R. Wallace et al., NIST, Special Publication 500-204, "High Integrity Software Standards and Guidelines" (1992); Dolores R. Wallace & Laura M. Ippolito, NBS, Special Publication 500-223, "A Framework for the Development and Assurance of High Integrity Software" (1994).

[59] See NBS, FIPS PUB 99, "A Framework for the Evaluation and Comparison of Software Development Tools" (1983); I. Trotter Hardy et al., NBS, Special Publication 500-14, "Software Tools: A Building Block Approach" (1977); Raymond C. Houghton, Jr., NBS, Special Publication 500-74, "Features of Software Development Tools" (1981); Herbert Hecht, NBS, Special Publication 500-82, "Final Report: A Survey of Software Tools Usage" (1981); Raymond C. Houghton, Jr., NBS, Special Publication 500-88, "Software Development Tools" (1982); Herbert Hecht, NBS, Special Publication 500-91, "The Introduction of Software Tools" (1982).

[60] See NBS, FIPS PUB 101, "Guideline for Lifecycle Validation, Verification, and Testing of Computer Software" (1983); NBS, FIPS PUB 132, "Guideline for Software Verification and Validation Plans" (1987); see also Martha A. Branstad et al., NBS, Special Publication 500-56, "Validation, Verification, and Testing for the Individual Programmer" (1980); W. Richards Adrion et al., NBS, Special Publication 500-75, "Validation, Verification, and Testing of Computer Software" (1981); Patricia B. Powell, NBS, Special Publication 500-98, "Planning for Software Validation, Verification, and Testing" (1982); Thomas J. McCabe, NBS, Special Publication 500-99, "Structured Testing" (1982); Dolores R. Wallace, NBS, Special Publication 500-136, "An Overview of Computer Software Acceptance Testing" (1986); Dolores R. Wallace et al., NIST, Special Publication 500-165, "Software Verification and Validation" (1989); Dolores R. Wallace & John C. Cherniavsky, NIST, Special Publication 500-180, "Guide to Software Acceptance" (1990); Wendy W. Peng & Dolores R. Wallace, NIST, Special Publication 500-209, "Software Error Analysis" (1993); Dolores R. Wallace et al., NIST, Special Publication 500-234, "Reference Information for the Software Verification and Validation Process" (1996); Dolores R. Wallace et al., NIST, Special Publication 500-235, "Structured Testing" (1996).

Two themes emerge from this literature. First, whereas older FIPS had been based on existing technologies or de facto practices, NIST increasingly sought to impose idealized notions of how software development ought to be practiced.[62] Second, much of that prescriptive guidance went disregarded by the larger software community.

A central pillar of NIST's efforts was the promotion of rigorous software documentation practices. NIST issued FIPS 38 in 1976, offering detailed and specific instructions on how to improve documentation at each stage of the software development life cycle.[63] NIST viewed documentation as being of primary importance and coding as a secondary task—but this stance was at odds with actual practices. Six years later, NIST convened a workshop on FIPS 38 and admitted that "[m]any software users are not familiar with these guidelines and standards." One commentator claimed that "[t]he fault ... is not in the guidelines but in the failure of code developers to consider documentation as an important function."[64] But the more common sentiment among attendees was that software documentation is not conducive to standardization, and that compliance with FIPS 38 did not produce useful results.[65]

NIST's efforts on software validation and verification followed a similar pattern of wishful standardization. NIST issued FIPS 101 in 1983, which recommended that software developers create a detailed validation, verification, and testing (VV&T) plan at the outset of the software life cycle.[66] NIST envisioned a top-down "waterfall" workflow where the VV&T plan would be completed during the initial project planning phase, well in advance of the code implementation phase.[67]

---

[61] See NBS, FIPS PUB 106, "Guideline on Software Maintenance" (1984); Martin & Osborne, NBS SP 500-106, supra note 56; James A. McCall et al., NBS, Special Publication 500-129, "Software Maintenance Management" (1985); Osborne, NBS SP 500-130, supra note 53.

[62] See Radack, supra note 24, at 194 (noting that the "character of the standards process for information technology has changed").

[63] FIPS 38, supra note 57, at 5, 13.

[64] See Neumann, NBS SP 500-94, supra note 57, at 5, 10, 40 (1982).

[65] Id. at 76–78 (collecting commentary); see also id. at 28 (noting that FIPS 38 is a set of flexible guidelines, not a standard, and that "[o]ffering the software documentor the license to completely depart from the guidelines vitiates the program that produced the guidelines").

[66] See FIPS 101, supra note 60, at 4 ("Validation determines the correctness of the final program or software with respect to the software requirements. Verification employs integrity and evolution checking to determine internal consistency and completeness.").

[67] See Powell, NBS SP 500-98, supra note 60, at 29 (noting that the preparation of the VV&T plan should be "completed early in the development phase"); see also FIPS 132, supra note 60, at 14 fig.1 (diagramming the waterfall lifecycle).

While meticulous planning is essential to conventional engineering projects, it proved paralyzing for software projects. Attempts to adhere to a plan-first-then-build approach (the waterfall method) typically resulted in cost overruns, delays, and overspecification of features that failed to fulfill user needs.[68] By contrast, commercial vendors were able to outcompete on both cost and quality by ignoring the government's software standards.[69] Conceding to on-the-ground realities, NIST later agreed that its validation and verification standards need not apply to "noncritical software."[70]

Even worse, FIPS 101 failed to require actual standardization. Instead, it suggested multiple soft factors to consider in constructing an appropriate VV&T plan, including "project needs and constraints," "the project's development approach," "overall schedule and budgets," and "size, complexity, and critical nature of the project."[71] Ultimately, NIST acknowledged, "[n]o single VV&T technique can guarantee correct, error-free software."[72]

This type of noncommittal language pervades NIST's software lifecycle standards.[73] On these higher-level aspects of software development, NIST offered its best notions of quality control, but there were

---

[68] See Defense Science Board, "Report on Acquiring Defense Software Commercially," supra note 53, at 33–34 (recommending removal of "any remaining dependence upon the assumptions of the 'waterfall' model" in military software standards).

[69] See Wallace et al., NBS SP 500-165, supra note 60, at 1 (noting that the approach advocated by NIST was "often ignored in today's highly competitive marketplace").

[70] See FIPS 132, supra note 60, at 9 ("For noncritical software, this standard does not specify minimum required V&V tasks[.]"). Even for critical software, vendors pushed successfully for relaxation of software standards during this same time period. See Choi, "Software as a Profession," supra note 8, at 578 (discussing softening of the DO-178 standard for avionics software).

[71] FIPS 101, supra note 60, at 19; see also Powell, NBS SP 500-98, supra note 60, at 27 (acknowledging that NIST's guidance "does not address the problem of how to select and configure specific techniques and tools for a specific project").

[72] FIPS 101, supra note 60, at 4.

[73] See, e.g., Fife, NBS SP 500-11, supra note 58, at 4 ("No technology or standard practice now exists that will surely prevent faulty design, logical errors, cost overrun, or late delivery for any software project."); Branstad et al., NBS SP 500-56, supra note 60, at 17 ("How do you know when you have tested enough? That's a fundamental question that unfortunately has no clear cut answer. ... [T]he amount of testing will depend upon the cost of an error."); Houghton, Jr., NBS SP 500-74, supra note 59, at 19 ("Software tools ... are not being effectively used in many Federal programming environments."); McCall et al., NBS SP 500-129, supra note 61, at 16 (comparing software maintenance to "trying to find 'a needle in the haystack.' ... There are no tried and true techniques that can immediately isolate the routine at fault."); Wallace et al., NBS SP 500-204, supra note 58, at xiv ("No standard can guarantee the safety of a particular software system. In other words, no one can ever say 'If a developer follows this standard, the system will be safe.'"); id. at 16

neither consensus practices nor objective metrics on which to ground such standards. Because of those basic gaps, the federal standards failed to provide clear guidance and failed to win general acceptance.

Data Encryption

NIST's most prominent set of FIPS have come in the area of cryptography. Viewed in isolation, the widespread adoption of NIST's encryption standards could be held up as a marker of success.

Several counterpoints, however, suggest that encryption is the exception that proves the agency's limitations. First, encryption algorithms are mathematical models that perform only a single function. The narrow mathematical basis makes it more straightforward to compare quantitative attributes such as encryption strength and efficiency. In that sense, an encryption algorithm is more akin to a conventional "weight or measure" than to a nebulous software quality metric. Second, that narrowness of scope greatly facilitates the adoption and enforcement of uniform standards. It allows NIST to run open competitions and select a consensus winner. Afterward, it also ensures that NIST can properly validate third-party implementations of the selected algorithm.[74] Third, other extrinsic factors have contributed to the outlier success of the encryption FIPS. Because of national security concerns, the National Security Agency (NSA) has regularly intervened in the FIPS process, simultaneously creating a channeling effect within the federal government while also attracting critical public attention to these FIPS in particular.

Soon after the Brooks Act of 1965, NIST recognized the need for a standard encryption algorithm for unclassified, sensitive information that would facilitate equipment interoperability, as well as avert Soviet spying.[75] NIST issued a public request for proposals in 1973 and 1974, which yielded a single candidate created by an IBM researcher.[76] In 1977, after collaborative review with the NSA, NIST published FIPS 46, also known as the Data Encryption Standard (DES).[77] There were vociferous critiques—later substantiated—that the NSA had altered the algorithm to reduce its strength.[78]

(observing that "[t]here is little agreement" on recommendations regarding standard practices for high integrity software).

[74] See NIST, Cryptographic Algorithm Validation Program, https://perma.cc/NY3D-REKX

[75] See Michael A. Froomkin, "The Metaphor Is the Key: Cryptography, the Clipper Chip, and the Constitution," *University of Pennsylvania Law Review* 143 (1995): 709, 735.

[76] See David P. Leech & Michael W. Chinworth, NIST, "The Economic Impacts of NIST's Data Encryption Standard (DES) Program," (2001) at ES-1.

[77] See Dorothy E. Denning, "The Data Encryption Standard: Fifteen Years of Public Scrutiny" (1990), https://perma.cc/GGC7-W4RN.

[78] See Nadiya Kostyuk & Susan Landau, "Dueling Over Dual_EC_DRBG: The Consequences of Corrupting a Cryptographic Standardization Process," *Harvard National Security Journal* 13 (2022): 224, 239 (citing Thomas Johnson, Center for Cryptological History, National Security Agency, *American Cryptology During*

Nevertheless, DES survived public scrutiny and enjoyed considerable popularity until 1998,[79] when it was finally deemed obsolete due to advances in computational capacity.[80]

The longevity of DES speaks not just to its strength but also to the halting pace at which new FIPS were approved. For two decades, the NSA repeatedly blocked NIST from issuing new FIPS that would have provided more secure cryptographic methods.[81] Eventually, the NSA developed its own proprietary algorithm,[82] and compelled NIST to publish the Escrowed Encryption Standard (EES) as FIPS 185, known colloquially as the Clipper Chip.[83] This time, the NSA's intent and interference were overt: The algorithm would operate in escrow, meaning the government would hold master keys that could decrypt messages on demand. Predictably, EES attracted heavy protest from the computer security community and beyond, mainly on the grounds that any back-door access makes an encryption algorithm inherently

the Cold War: 1945–1989; Book III: Retrenchment and Reform: 1972–1980 (1998), at 232 https://perma.cc/B2AT-QBCK). This suspicion of government-controlled cryptography resulted in the development of public key cryptography, which eliminated dependence on a trusted key exchange. See Johnson, supra, at 234 (explaining that Hellman, the co-creator of the Diffie-Hellman algorithm, "had been one of the leading opponents of DES, for the very reason that he distrusted NSA's hand in the algorithm"). More secure cryptographic techniques such as Diffie-Hellman and its successor, RSA, were never adopted as FIPS; instead, the NSA worked "diligently" to suppress publication of such methods. See id. at 235; Susan Landau, "Under the Radar: NSA's Efforts to Secure Private-Sector Telecommunications Infrastructure," *Journal of National Security Law and Policy* 7 (2014): 411, 421.

[79] See Landau, supra note 78, at 418 ("Many believed that DES's design and short key size made the algorithm potentially breakable by the NSA, but in fact, the algorithm has stood the test of time."); Johnson, supra note 78, at 239 ("By the early 1990s [DES] had become the most widely used encryption algorithm in the world.").

[80] See Kostyuk & Landau, supra note 78, at 241; Miles E. Smid, "Development of the Advanced Encryption Standard," *Journal of Research of the National Institute of Standards and Technology* 126 (2021), art. 126024, at 2.

[81] See Landau, supra note 78, at 421.

[82] See Froomkin, supra note 75, at 753 (describing the NSA's ten-year effort to develop SKIPJACK, the classified algorithm at the heart of the Clipper Chip).

[83] See id. at 778–89 (detailing extensive cooperation between NIST and the NSA regarding the Clipper Chip); Kostyuk & Landau, supra note 78, at 240 (noting that NIST was "quite deferential to NSA").

insecure.[84] Although the Clipper Chip was endorsed heavily by the U.S. government, it withered on the vine.[85]

In the aftermath of the Clipper Chip episode, NIST still needed a successor candidate to replace the aging DES algorithm. NIST launched a public competition for a new Advanced Encryption Standard (AES) and devoted substantial resources to resuscitating goodwill among the cryptographic community.[86] Recognizing that a standard developed by the U.S. intelligence community would not be acceptable,[87] NIST persuaded the NSA to take a back seat.[88] The competition was a success: The AES development process was praised for being fully open and transparent.[89] NIST published the new standard in 2001 as FIPS 197; unlike the Clipper Chip, AES was widely adopted.[90]

Even though the NSA now seemed to accept the need for strong encryption standards, security researchers continued to suspect that the NSA was tampering with NIST's standards. In 2006, NIST approved the use of DUAL_EC_DRBG, a new random number generator, for generating encryption keys. Several commentators raised suspicions that the new algorithm was flawed and that the NSA had

[84] See Landau, supra note 78, at 423 ("It is hard to imagine a more negative reaction to Clipper than the one that ensued."); Kostyuk & Landau, supra note 78, at 245–46 (2 comments in favor and 318 opposed); Smid, supra note 80, at 4 (noting that there were "immediate" concerns regarding the secrecy of the algorithm, and that "just having the escrow feature weakened the security of the encryption system"); Froomkin, supra note 75, at 772 (noting that NIST received hundreds of critical comments but rejected them "on the disingenuous grounds that because the standard was entirely voluntary, it could cause no harm"); see also Kostyuk & Landau, supra note 78, at 246 ("From the outside, it looked as if NIST was not listening to public input and was heading towards cryptographic standards that provided security but not necessarily privacy. Few knew that NIST had actually pressed for the industry-favored technique but had been overruled [by the NSA].").

[85] See Landau, supra note 78, at 423.

[86] See id. at 427.

[87] See Smid, supra note 80, at 5 ("The key escrow program demonstrated that an algorithm designed, evaluated, and proposed as a standard by the U.S. government would likely have a difficult time achieving consensus.").

[88] See id. at 9 (noting serious concerns that a NSA submission might create a perception that "NIST led a sham competition," and that "[i]n the end, NSA chose not to submit a candidate algorithm").

[89] See Kostyuk & Landau, supra note 78, at 236 (describing the general perception of NIST as an "'honest broker' that favors neither a particular company nor a country"); id. at 248 ("The cooperative relationship that NIST had forged with the cryptographic research community during the AES competition led to the agency assuming a leadership role in the development of internationally adopted cryptographic standards.").

[90] See id. at 247–48 (calling NIST's process "a model of openness and transparency" and estimating the economic benefit of AES at $250 billion).

used NIST to slip the flawed algorithm into all FIPS-certified cryptographic systems.[91] Those suspicions were substantiated in 2013 when Edward Snowden leaked classified documents to the public.[92] NIST responded by immediately recommending against use of the DUAL_EC_DRBG algorithm and subsequently rescinding its approval.[93] Some observers argue that NIST's strong response helped it successfully weather the storm and maintain good relations with the crypto community. [94] But more skeptical voices claim that NIST failed to heed much earlier warnings and knowingly approved a flawed standard.[95]

Despite these stumbles, NIST continues to play a critical convening and standard-setting role within the cryptographic community. Defenders of the agency have argued that NIST's influence stems from its power to issue FIPS, which generates a rallying effect, as well as from its unique reputation as a neutral, trusted arbiter.[96] But if NIST's organizational power and reputation were enough to ensure the success of FIPS, then one would expect NIST to maintain FIPS in more areas beyond cryptography. Instead,

---

[91] See Bruce Shneier, "Did NSA Put a Secret Backdoor in New Encryption Standard?" *Wired*, Nov. 15, 2007.

[92] See Kostyuk & Landau, supra note 78, at 235–37, 247; see also Nicole Perlroth et al., "N.S.A. Able to Foil Basic Safeguards of Privacy on Web," *New York Times*, Sept. 5, 2013 (noting that classified memos "appear to confirm" that the NSA deliberately weakened the international encryption standard adopted in 2006 by NIST); Bruce Shneier, "The US Government Has Betrayed the Internet. We Need to Take It Back," *Guardian*, Sept. 5, 2013 ("[T]he NSA has undermined a fundamental social contract. ... [T]he US has proved to be an unethical steward of the internet.").

[93] See Kostyuk & Landau, supra note 78, at 250.

[94] See id. at 259–60 (opining that the Dual_EC_DRBG situation was "something 'that happened' to NIST, rather than something NIST caused"); id. at 267 (observing that "[n]o other nation has a federal agency involved in such activities" and that the fact that FIPS cryptographic standards are mandatory for government procurement purposes "provides a natural market for the standards"). Although the authors acknowledge that private companies such as Google have been able to establish cryptographic standards for the industry, they argue that such private entities lack the organizational capacity and the trust to be a "primary developer" of such standards. Id. at 276–79.

[95] See, e.g., Daniel J. Bernstein, "NIST's Cryptographic Standardization Process," *Cr.yp.to Blog*, April 11, 2014, https://perma.cc/MR9Y-HTLV (complaining that NIST is publishing standards at a "reckless pace").

[96] See Kostyuk & Landau, supra note 78, at 265, 269, 277 (noting that "only NIST can create a FIPS," which are "mandated for equipment sold for U.S. government use, which creates a large follow-on effect of widespread global adoption"); id. at 276–77 ("NIST ensures that everyone's voice is heard, and the process includes a transparent and rigorous peer review. Members of the cryptographic community, especially academics, are happy with this arrangement."). The authors argue further that private entities cannot mandate compliance via law, and that foreign governments are less capable of establishing internationally trusted cryptographic standards.

cryptography has emerged as the exceptional case. A more plausible explanation is that there is something unique about the cryptographic field that makes it more amenable to standardization than other aspects of software.

Modern Computing Standards

The many shortfalls of NIST's software standardization efforts, despite outsized investment over multiple decades, weakened NIST's claim to authority in the area. Commercial systems developed by private industry consistently outperformed software developed for government. It became increasingly difficult to justify expenditures for NIST to develop and maintain federal software standards that were separate from those used by private vendors.

Beginning in the mid-1990s, NIST took a hiatus from its prior efforts to lead the standardization of computing protocols. The formal impetus for this shift was the enactment of the National Technology Transfer and Advancement Act of 1995, which declared that "all Federal agencies and departments shall use technical standards that are developed or adopted by voluntary consensus standards bodies."[97] But this pivot was the culmination of a longer-term retreat by the U.S. government from competing directly with private developers of off-the-shelf software.

With the notable exception of encryption and information security standards,[98] NIST withdrew its FIPS and did not issue new ones.[99] Although NIST remained an active member of private standard-setting bodies, its official duties in computing and software policy diminished to encryption and information security, and to one-off studies authorized by Congress on topics such as voting machines, identity management systems, and smart electrical grids.[100]

---

[97] National Technology Transfer and Advancement Act of 1995, Pub. L. No. 104-113 § 12(d), 110 Stat. 775, 783 (1996).

[98] See Federal Information Security Management Act (FISMA) of 2002, Pub. L. No. 107-347 § 303, 116 Stat. 2899, 2957 (directing NIST to develop standards and guidelines "for providing adequate information security for all agency operations and assets," other than for national security systems), amended and superseded by Federal Information Security Modernization Act of 2014, Pub. L. No. 113-283, 128 Stat. 3073 (2014); Computer Security Act of 1987, Pub. L. No. 100-235, 101 Stat. 1724 (1987). See generally Eric P. Roberson, "'Adequate' Cybersecurity: Flexibility and Balance for a Proposed Standard of Care and Liability for Government Contractors," *Federal Circuit Bar Journal* 25 (2016): 641, 673–77.

[99] See NIST, NIST Technical Series Publication List: FIPS, https://perma.cc/UVK7-58UY.

[100] See, e.g., USA PATRIOT Act of 2001, Pub. L. No. 107-56 § 403(c), 115 Stat. 272, 344 (2001) (codified at 8 U.S.C. § 1379); Enhanced Border Security and Visa Entry Reform Act of 2002, Pub. L. No. 107-173 § 202, 116 Stat. 548 (2002) (codified at 8 U.S.C. § 1722); Help America Vote Act of 2002, Pub. L. No. 107-252, § 231, 116 Stat. 1684 (2002) (codified at 52 U.S.C. § 20971); Cyber Security Research and

That interregnum ended in 2013, when President Obama signed Executive Order 13636, tasking NIST with developing a cybersecurity framework to reduce cyber risks to critical infrastructure.[101] The order did not direct NIST to develop or certify new FIPS; instead, it specified that NIST's work should "incorporate voluntary consensus standards and industry best practices to the fullest extent possible."[102] Nevertheless, the clear intent was to establish "performance goals" that would function much like a federal standard in terms of establishing a minimum set of criteria for compliance.[103]

Since then, each administration in the White House has used executive orders to bypass congressional deadlock and to launch major new initiatives on cybersecurity, secure software development, and artificial intelligence. What ties together these initiatives is not only their ambitious, sweeping scope, but also the recommitment to NIST leadership to develop federal software standards.

Yet, little has changed in the interim to suggest that NIST is newly capable of producing effective consensus standards to regulate software quality. Instead, NIST has subtly adapted the executive order directives by embracing an open-tent "framework" approach that repudiates the conventional standardization model. These new frameworks invite universal participation and avoid policing the bounds of compliance or noncompliance. The big takeaway is that NIST's new frameworks are not uniform standards or measures of anything. Therefore, it is quite nonsensical to treat them as any type of threshold for determining legal liability.

Development Act, Pub. L. No. 107-305 § 8, 116 Stat. 2375 (2002) (codified at 15 U.S.C. § 7406); Energy Independence and Security Act of 2007, Pub. L. No. 110-140 § 1305, 121 Stat. 1787 (2007) (codified at 42 U.S.C. § 17385).

[101] See Executive Order 13636, supra note 1. This executive order was subsequently supported by legislation. See Cybersecurity Enhancement Act of 2014, Pub. L. No. 113-274 § 502, 128 Stat. 2971, 2986 (directing NIST to "ensure coordination of Federal agencies engaged in the development of international technical standards related to information system security"). See generally Melanie Teplinsky, "Fiddling on the Roof: Recent Developments in Cybersecurity," *American University Business Law Review* 2 (2013): 225, 300 (providing background on the executive order, which arose in reaction to Congress's failure to pass the Lieberman-Collins Cybersecurity Act of 2012). Previously, the White House had already begun to involve NIST in developing standards and guidelines for related aspects such as cloud computing. See NIST, "NIST Helps Accelerate the Federal Government's Move to the Cloud" (press release), June 9, 2010, https://perma.cc/2YXN-LDEQ.

[102] Executive Order 13636, supra note 1, at § 7(a).

[103] Id. § 7(d).

Cybersecurity Framework

Within one year of President Obama's 2013 executive order, NIST released its Cybersecurity Framework in 2014, with a minor update in 2018.[104] Perhaps because of the short turnaround demanded, the document closely resembles the general "risk management framework" NIST had already developed pursuant to the Federal Information Security Management Act of 2002 (FISMA).[105] While FISMA applies only to the federal government's own software procurement practices, the Cybersecurity Framework is intended for a broader audience beyond the federal government.

FISMA had already attracted criticism for being ineffective at managing cybersecurity risk.[106] According to some commentators, the risk assessment approach encouraged a "checklist" or "paperwork drill" mentality among federal agencies.[107] Enforcement was lax,[108] and commentators worried that the FISMA approach ignored the root problem of software quality.[109] Nor did the FISMA framework

[104] See NIST, "NIST Releases Cybersecurity Framework Version 1.0" (press release), Feb. 12, 2014, https://perma.cc/L73K-KWG6. NIST released an updated version 1.1 in 2018. NIST, "NIST Releases Version 1.1 of Its Popular Cybersecurity Framework" (press release), April 16, 2018, https://perma.cc/8D2N-WG6A. NIST released version 2.0 as a discussion draft in 2023. NIST, "NIST Drafts Major Update to Its Widely Used Cybersecurity Framework" (press release), Aug. 8, 2023, https://perma.cc/NJ26-Y8R7.

[105] See Gary Stoneburner et al., NIST, Special Publication 800-30, "Risk Management Guide for Information Technology Systems" (2002), superseded by NIST, Special Publication 800-30 Rev. 1, "Guide for Conducting Risk Assessments" (2012); Ron Ross et al., NIST, Special Publication 800-53, "Recommended Security Controls for Federal Information Systems" (2005); see also NIST, "Federal Information Security Modernization Act (FISMA) Background," https://perma.cc/Y3HX-3V96.

[106] See Daniel M. White, Note, "The Federal Information Security Management Act of 2002: A Potemkin Village," *Fordham Law Review* 79 (2010): 369, 377 (collecting criticisms that FISMA is (1) difficult to implement, (2) a mere "paperwork exercise," and (3) not addressing software quality issues); see also Chelsea C. Smith, Comment, "Hacking Federal Cybersecurity Legislation: Reforming Legislation to Promote the Effective Security of Federal Information Systems," *National Security Law Journal* 4 (2016): 345, 370 (noting that "FISMA is a 'well-intentioned but fundamentally flawed tool' because it provides a mechanism for information security planning as opposed to serving as an effective method for actually measuring and improving information security").

[107] See White, supra note 106, at 380–82; Smith, supra note 106, at 371.

[108] See Smith, supra note 106, at 374.

[109] See White, supra note 106, at 383–87.

improve actual cybersecurity performance: Observers noted that the number of security incidents increased dramatically over the implementation period.[110]

In practice, the Cybersecurity Framework operates as an auditing manual that teaches organizations how to perform a self-assessment report. That self-assessment establishes the organization's current baseline of protective measures ("Current Profile"), along with an aspirational "Target Profile" that the organization hopes to attain in the future.

The Framework enumerates a set of "Functions" that could help reduce one's risk exposure.[111] At a high level, those Functions are (1) identifying cybersecurity risk, (2) protecting against potential attacks, (3) detecting cybersecurity events, (4) responding to such incidents, and (5) recovering from harms caused thereby.[112] Each Function is broken down into many smaller components and subcomponents. Each entity is free to select which subcomponents to include in or exclude from its Target Profile.

As an illustrative example, the "Respond" function includes subcomponent "RS.MI-3" advising that "Newly identified vulnerabilities [should be] mitigated or documented as accepted risks."[113] Each subcomponent then refers out to a number of acceptable external standards such as NIST Special Publication 800-53. Here, the RS.MI-3 function refers to three controls in the SP 800-53 document: (1) continuous monitoring of "*organization-defined metrics*" on "*organization-defined frequencies*"; (2) reviewing and updating risk assessments on an "*organization-defined frequency*"; and (3) remediating legitimate vulnerabilities on "*organization-defined response times*" in accordance with an "*organizational assessment of risk*."[114] The italicized terms are key values that the Framework prompts each organization to set for itself. The open-ended nature of these controls characterizes the vast majority of the Framework, with the exception of certain cryptographic items.

[110] See Smith, supra note 106, at 370–71 (citing "a more than 1,120 percent increase from FY 2006 through FY 2014" in security incidents); White, supra note 106, at 382 (citing a 250 percent increase between 2007 and 2009).

[111] For a similar overview of the Cybersecurity Framework, see McGeveran, supra note 4, at 1161.

[112] Version 2.0 proposes adding "govern" as a sixth function, which addresses how an organization can prioritize and direct the execution of the other five functions in its overall cybersecurity strategy. See NIST, "NIST Drafts Major Update," supra note 104.

[113] NIST, Cybersecurity Framework v.1.1, at 43 (2018) (pointing to controls CA-7, RA-3, and RA-5 in NIST SP 800-53 Rev.4).

[114] See NIST, Special Publication 800-53 Rev. 4, "Security and Privacy Controls for Federal Information Systems and Organizations," (2021), at App. F-60, F-152, F-153 (italics in original) (describing controls CA-7, RA-3, and RA-5).

There are no minimum requirements for compliance with the Framework. For each sub-function, the Framework defines four "Tiers" of cybersecurity readiness: (1) partial, (2) risk informed, (3) repeatable, and (4) adaptive. At each progressive tier, an organization is expected to maintain increasingly formal policies and procedures. Organizations are encouraged to move toward higher tiers—but only if the cost-benefit balance is reasonable.[115]

NIST emphasizes the flexible nature of the document: "The Framework is not a one-size-fits-all approach .... Organizations can determine activities that are important to critical service delivery and can prioritize investments to maximize the impact of each dollar spent. ... The decision about how to apply [the Framework] is left to the implementing organization."[116] NIST resists the idea that one can be "compliant" with the Cybersecurity Framework, since it is merely a planning document that varies according to each organization's respective strategies and goals.

The Cybersecurity Framework has become a popular document, [117] largely due to its flexibility.[118] Nevertheless, persistent questions linger as to its substance. Many organizations appreciate that the Framework does not require them to change their practices at all, either because they are already working within another risk management framework, or because their cost-benefit analysis suggests changes would be unreasonable. Adopting a big-tent stance means that all organizations can nominally claim to be "implementing" or "in compliance with" the Framework. What is missing from the Framework is any quantifiable or standardized performance metrics that would allow meaningful comparison across organizations of standard versus substandard care.[119]

[115] NIST, Cybersecurity Framework v.1.1, supra note 113, at 8–9 ("While organizations identified as Tier 1 (Partial) are encouraged to consider moving toward Tier 2 or greater, Tiers do not represent maturity levels. ... Successful implementation of the Framework is based upon achieving the outcomes described in the organization's Target Profile(s) and not upon Tier determination.").

[116] Id. at vi.

[117] See U.S. Chamber of Commerce, "Comment on NIST Cybersecurity Request for Information," April 25, 2022, https://perma.cc/A2RN-GCFU (observing that "[b]road swaths of the business community support the popular Cybersecurity Framework").

[118] See Lawrence A. Gordon et al., "Integrating Cost-Benefit Analysis Into the NIST Cybersecurity Framework via the Gordon-Loeb Model," *Journal of Cybersecurity* 6 (2020), at *2 (noting that the Cybersecurity Framework is "intentionally broad and flexible," and "lacks specificity and thus is ambiguous in terms of guidance").

[119] See Jim Dempsey, "Cybersecurity Regulation: It's Not 'Performance-Based' If Outcomes Can't Be Measured," *Lawfare*, Oct. 6, 2022 (describing NIST's framework as a "management-based approach" and calling for greater focus on performance-based standards that impose objectives with "measurable outcomes"). But see Cary Coglianese, "The Limits of Performance-Based Regulation," *University of*

Software Development Framework

In 2021, President Biden expanded NIST's responsibilities and revived its role in issuing software development standards.[120] Executive Order 14028 directed NIST to perform a litany of new tasks, including (1) define "critical software," (2) issue guidance on security measures for critical software, (3) issue guidance on enhancing software supply chain security, (4) issue guidance on software testing, and (5) define "secure software development practices."[121]

NIST has begun to release responsive publications, the most notable of which has been the Secure Software Development Framework (SSDF).[122] Heavily shaped by Microsoft interests, the SSDF echoes Microsoft's Security Development Lifecycle[123] without requiring fealty to any one model. The SSDF aims to be a flexible document that does not prescribe tools, techniques, or mechanisms.[124] NIST explains that the SSDF can be used "by organizations in any sector or community, regardless of size or cybersecurity sophistication" and can be used "for any type of software development, regardless of technology, platform, programming language, or operating environment."[125] To be sure, the SSDF effort was led by Microsoft in order to head off more rigid alternatives. Many commentators view the SSDF as a promising direction that builds on Microsoft's decades of experience improving its own software development practices. At the same time, Microsoft's own software security practices have continued to come under fire after numerous high-profile incidents in recent years.[126]

Michigan Journal of Law Reform 50 (2017): 525, 553–62 (enumerating multiple problems with performance standards and, in particular, the risks of mismatch with root policy goals, tunnel vision, and gaming of the system).

[120] See Executive Order 14028, supra note 1.

[121] Id. §§ 4(g), 4(i), 4(e), 4(r), 4(u).

[122] See Murugiah Souppaya et al., NIST, Secure Software Development Framework (SSDF) Version 1.1 (2022), https://doi.org/10.6028/NIST.SP.800-218; see also NIST, "Definition of Critical Software Under Executive Order (EO) 14028" (2021), https://perma.cc/N2JC-UNF8; NIST, "Security Measures for "EO-Critical Software" Use Under Executive Order (EO) 14028" (2021), https://perma.cc/6MZS-J83K.

[123] See generally Michael Howard & Steve Lipner, *The Security Development Lifecycle 27–37* (2006) (offering a short history of the origins of Microsoft's Security Development Lifecycle).

[124] See NIST, SSDF, supra note 122, at 2.

[125] Id. at vi.

[126] See Ellen Nakashima et al., "Chinese Hackers Breach Government Email Accounts Through Microsoft Cloud," *Washington Post*, July 12, 2023 (detailing several recent incidents and suggesting Microsoft's security failures are "a habit, not an anomaly").

The SSDF comprises four sets of recommended practices. The first two sets are reasonably straightforward to implement: Organizations should (1) "prepare" their workforce and work environments in accordance with the security requirements and (2) "protect the software" against any tampering. The latter two sets represent the core challenge: Organizations should (3) "produce well-secured code" and (4) "respond to vulnerabilities."

Focusing on the third set, the production of secure code involves nine "tasks" that effectively perform four major functions. The first function involves the design stage, to ensure that the software specifications incorporate appropriate security requirements. The second is the implementation stage. The SSDF exhorts users to "follow all secure coding practices that are appropriate to the development languages and environment to meet the organization's requirements," for example, by validating all inputs and outputs, avoiding unsafe function calls, using automated tools, and performing code reviews.[127] When in doubt, the SSDF suggests reusing existing code, which is preferred to generating new code. Third, secure software developers should perform software testing for vulnerabilities and for verification of security requirements. Fourth, the software should be configured properly with secure defaults.

Design, implementation, and testing are the critical core of any secure software development framework. One might assume that an effective standard would define effective guardrails for these aspects. The SSDF collects a set of best practices, but it does not dictate how to perform those tasks or whether any are required. The SSDF purports to focus on "outcomes," rather than mandating specific technical approaches.[128] Yet, like the Cybersecurity Framework, the SSDF similarly elides any definition of what outcomes should be measured, and how. To be sure, the SSDF is hardly an outlier: It cites numerous standards issued by private organizations, all of which adopt the same stance of endorsing free exercise of discretionary judgment by software developers.[129]

For the fourth set, the SSDF asks organizations to respond to vulnerabilities in a reasonable, cost-effective way. Again, the SSDF does not attempt to set firm requirements regarding timing of fixes,

[127] NIST, SSDF, supra note 122, at 13.

[128] Id. at vi.

[129] Some examples include International Electrotechnical Commission (IEC), IEC 62443-4-1, Secure Product Development Lifecycle Requirements (2018), https://perma.cc/2J4N-PNNZ; Microsoft, Security Development Lifecycle (2021), https://www.microsoft.com/en-us/securityengineering/sdl/; Business Software Alliance, The BSA Framework for Secure Software: A New Approach to Securing the Software Lifecycle, Version 1.1 (2020), https://perma.cc/AC7N-F668; Software Assurance Forum for Excellence in Code, Fundamental Practices for Secure Software Development: Essential Elements of a Secure Development Lifecycle Program, 3rd ed. (2018), https://perma.cc/NAA3-2EG7.

warning to those affected, or product recall.[130] Nor does it grapple with the complexities of how to design and deploy safe software patches, which can introduce new problems even as they fix old ones.

AI Framework

The White House also expanded NIST's role into other adjacent areas. In 2019, President Trump issued Executive Order 13859, directing NIST to develop "technical standards and related tools in support of reliable, robust, and trustworthy systems that use AI technologies."[131] President Biden followed up in 2023 with Executive Order 14110, commanding NIST to produce additional guidelines, standards, and best practices on developing, deploying, and red-team testing "safe, secure, and trustworthy AI systems."[132]

NIST's response remains in early stages.[133] In January 2023, NIST released the first version of its AI Risk Management Framework.[134] NIST defines the central goal of the AI Framework as "trustworthiness,"[135] which is its way of asserting that the relevant risks have been appropriately

[130] Cf. Andrea M. Matwyshyn, "Hidden Engines of Destruction: The Reasonable Expectation of Code Safety and the Duty to Warn in Digital Products," *Florida Law Review* 62 (2010): 109; Matthew T. Wansley, "The Auto Safety Revolution," *Emory Law Journal* 73 (forthcoming 2024), https://ssrn.com/abstract=4190688 (describing use of "recalls" to force software developers to either fix their code or to restrict where it can operate).

[131] See Executive Order 13859, supra note 2.

[132] See Executive Order 14110, supra note 2.

[133] See NIST, "A Report to Congress: Steps to Implement Recommendations Regarding 'U.S. Leadership in Artificial Intelligence (AI): A Plan for Federal Engagement in Developing Technical Standards and Related Tools'" (2022), https://perma.cc/BJ4C-4EZG.

[134] NIST, AI Risk Management Framework 1.0, https://nvlpubs.nist.gov/nistpubs/ai/nist.ai.100-1.pdf; see also NIST, "NIST Risk Management Framework Aims to Improve Trustworthiness of Artificial Intelligence" (press release), Jan. 26, 2023, https://perma.cc/YQG8-CDPC (noting that NIST worked on the framework for 18 months).

[135] See AI Framework, supra note 134, at 10–11 ("Approaches which enhance AI trustworthiness can also contribute to a reduction of AI risks. This Framework articulates the following characteristics of trustworthy AI, and offers guidance for addressing them. Trustworthy AI is: valid and reliable, safe, fair and bias is managed, secure and resilient, accountable and transparent, explainable and interpretable, and privacy-enhanced."); Brian Stanton & Theodore Jensen, NIST, "Trust and Artificial Intelligence" (2020), at 7 (defining AI trustworthiness as the "ability to perform as and when required"); see also Jeannette M. Wing, "Trustworthy AI," *Communications of the ACM* (October 2021), at 64, 65 (describing the progression at the National Science Foundation (NSF) from Trusted Computing (2001) to Cyber Trust (2004), Trustworthy Computing (2007), and Secure and Trustworthy Cyberspace (2011), and explaining that "support for

managed.[136] Left indeterminate is whose trust should be prioritized.[137] The document acknowledges that the management of AI risks usually involves trade-offs and "difficult decisions," but ultimately declines to make any firm choices, calling trustworthiness a "social concept" that "depends on an AI actor's particular role within the AI lifecycle."[138]

The AI Framework adheres to NIST's general risk management methodology of classifying risks, implementing protective steps, and documenting remedial plans. Thus, the AI Framework outlines four functions: govern, map, measure, and manage. First, the "govern" function puts in place a documentation regime that "cultivates a culture of risk management."[139] Second, the "map" function serves as an impact assessment to identify the potential risks of an AI system. Third, NIST asks organizations to "measure" each of the identified AI risks using quantitative, qualitative, or other techniques. Notably, however, the AI Framework observes that "[h]uman judgment must be employed when deciding on the specific metrics related to AI trustworthy characteristics and the precise threshold values for their related metrics."[140] Fourth, organizations should "manage" any AI risks to minimize the likelihood of system failures and negative impacts.

As with NIST's other standardization efforts, the effectiveness of the AI Framework will depend on how specific and enforceable the guidance will be. For example, NIST claims it is seeking to develop a range of evaluation tools such as (1) data sets in standardized formats for training, validation, and testing; (2) standardized knowledge representation tools that could promote interoperability of AI systems; (3) case

---

research in trustworthy computing now spans multiple directorates at NSF and engages many other funding organizations").

[136] Cf. Margot E. Kaminski, "The Developing Law of AI: A Turn to Risk Regulation," Digital Social Contract Paper Series, *Lawfare* (April 2023), at 3 (criticizing the risk regulation approach for "presum[ing] that the technology need only be tweaked at the edges").

[137] This renewed interest in "trustworthiness" recalls an older line of Orwellian critique of trusted computing, wherein "trust" becomes doublespeak for surveillance and control of computer users when they are perceived to be the source of risk. See Chad Woodford, Note, "Trusted Computing or Big Brother? Putting the Rights Back in Digital Rights Management," *University of Colorado Law Review* 75 (2004): 253, 279; Steven J. Vaughan-Nichols, "How Trustworthy Is Trusted Computing?" *Computer* (March 2003), at 18, 20 (noting criticisms that "trusted computing" gives vendors too much power and "would take away freedom by making decisions about data and applications that typically have been left to users").

[138] NIST, AI Framework, supra note 134, at 12–13.

[139] Id. at 21–22 ("Documentation can enhance transparency, improve human review processes, and bolster accountability in AI system teams.").

[140] Id. at 11.

studies; (4) benchmarks; (5) testing methodologies; (6) quantitative metrics; (7) AI testbeds; and (8) auditing tools.[141] Such tools, if completed, offer real promise of meaningful standardization.

But much about the AI Framework remains underdetermined. The AI Framework is voluntary.[142] It is neither "a checklist" nor "an ordered set of steps."[143] Evaluations of its effectiveness are unknown and "will be part of future NIST activities."[144] Moreover, its scope is extraordinarily broad, even relative to other NIST frameworks. The AI Framework enumerates seven expansive categories of AI risk: the trustworthy AI system must be (1) valid and reliable, (2) safe, (3) fair and unbiased, (4) secure and resilient, (5) explainable and interpretable, (6) privacy-enhanced, and (7) accountable and transparent. Thus far, NIST has held workshops and published draft reports on two aspects: bias and explainability.[145] Far from narrowing the search for AI standards, these early documents ruminate that trustworthy AI is a "socio-technical" concept, and that identifying measurement techniques remains an "emerging area."[146]

NIST'S SOFTWARE UN-STANDARDS

For those seeking a clearer software liability standard, there is obvious appeal to the centralized agency model. A single authority could marshal expertise from the field, declare a consensus set of technical

[141] See NIST, "U.S. Leadership in AI: A Plan for Federal Engagement in Developing Technical Standards and Related Tools" (2019), at 13–15; see also NIST, "A Report to Congress," supra note 133, at 7–10 (describing preliminary efforts by NSF to fund partnerships to develop such tools). But see id. at 4 (explaining the obstacle that "several agencies indicated their AI standards needs, and activities will be driven by operational needs and requirements as their agencies have very diverse AI standards-related needs").

[142] NIST, AI Framework, supra note 134, at 2; see also Kaminski, supra note 136, at 12 (comparing NIST's approach as being soft law like Singapore's Model AI Governance framework, rather than hard law like the EU's AI Act).

[143] NIST, AI Framework, supra note 134, at 20.

[144] Id. at 19.

[145] See Reva Schwartz et al., NIST, Special Publication 1270, "Towards a Standard for Identifying and Managing Bias in Artificial Intelligence" (2022); P. Jonathon Phillips et al., NIST, "Four Principles of Explainable Artificial Intelligence" (2021).

[146] NIST, "Towards a Standard for Identifying and Managing Bias," supra note 145, at 11 ("Socio-technical approaches in AI are an emerging area ... . Developing scientifically supportable guidelines to meet socio-technical requirements will be a core focus."); see also NIST, "Four Principles of Explainable Artificial Intelligence," supra note 145, at 22 (commenting that "understanding general principles that drive human reasoning and decision making may prove to be highly informative for the field of explainable AI" and that "[c]onsidering these human factors within the context of explainable AI has only just begun").

standards, and enforce uniform compliance with those standards.[147] Those centrally promulgated technical standards could then form the basis of a judicially enforceable liability rule.[148] Although many commentators have called for the creation of a new federal agency to undertake this role, the White House has already been promoting NIST as the incumbent candidate. And in many ways, NIST is a uniquely apt agency to be the standard-bearer, not least because of its well-established reputation as a nonpoliticized, scientific body.

Nevertheless, a close examination of NIST's work shows that there are at least three reasons to doubt any centralized agency's ability to cut the Gordian knot. First, the *historical record*: NIST has already produced thorough and voluminous guidance on software standards, much of it with disappointing impact. Even if another agency were to duplicate those extensive efforts, at substantial cost, it is unlikely to discover some dramatic new insight that NIST has not already considered.

Second, NIST's modern pivot to *self-governance frameworks* shows that NIST has shied away from the active steering role the White House had hoped it would take. Instead, NIST has emphasized flexibility over uniformity, documentation over metrics, and inclusion over enforcement. That deliberate abdication signals NIST's conclusion that a more forceful agency approach would be ineffective or detrimental.

Third, *institutional competence*: The success of the agency model depends on the comparative advantage that agencies have at marshaling scientific or technical expertise to produce policy consensus. Yet, NIST's inability to produce strong software standards is a symptom of a fundamental gap in the science of the field. As long as that gap persists, it is unrealistic to expect the agency model to locate an expert consensus that does not exist.

Ultimately, NIST's experience reveals an important lesson for the software liability literature. As NIST has moved from low-level hardware standards to higher-level abstractions of software quality, NIST's guidance has turned steadily away from discrete specifications in favor of a syncretic approach that allows many heterodoxies to coexist. If NIST is correct that no expert consensus exists, then the implication is that other agencies will similarly struggle to establish clear software standards. Concomitantly, courts and legislatures will need to look elsewhere to construct effective liability rules.

---

[147] See Choi, "Institutional Choice," supra note 8.

[148] See, e.g., Derek E. Bambauer, "Cybersecurity for Idiots," *Minnesota Law Review Headnotes* 106 (2021): 172, 175 (proposing a negligence per se approach in which generalist regulators establish "regulatory floors by specifying conduct that automatically generates liability"); Shackleford, supra note 4, at 103, 105 (noting that "U.S. states have become active laboratories for cybersecurity policymaking in the absence of federal leadership" and that "there is a general trend across many states to require firms to implement 'reasonable' cybersecurity best practices without clearly defining what those entail").

Historical Record

In many ways, the turn to NIST is déjà vu all over again. Close review of NIST's historical FIPS and special publications reveal an agency that engaged carefully and thoughtfully with the expertise in the field. Nevertheless, those centrally coordinated efforts were unable to consolidate and standardize software development practices. That past experience ought to give policymakers pause when wagering whether an agency like NIST can achieve better results the second time around.

There are at least three considerations that bear on why NIST failed to achieve greater success with software standards at the peak of its powers during the 1980s. The first factor is that the standardization process was too slow and costly relative to the pace of software innovation. Initially, NIST believed that the force of federal mandate would be sufficient to compel widespread adoption of its standards. Accordingly, NIST was very deliberate in its development and review of new proposed standards, typically taking several years or more to move from initial study to final issuance. For example, NIST invested heavily in promulgating detailed forms and standards for the COBOL programming language, with the expectation that COBOL would dominate for many years to come. Instead, NIST was caught off guard when software vendors and purchasers overwhelmingly preferred newer, nonstandard languages such as C. Even for federally approved languages, NIST was mostly unable to prevent programmers from using nonstandard features, rendering NIST's standards obsolete.

Second, NIST sought to standardize software activities at too high an abstraction. To be sure, NIST excelled at narrowly scoped standards such as data recording formats, ASCII and geographical codes, and encryption standards. But as the scope broadened to higher-level generalities, NIST was unable to demonstrate a clear payoff that adherence to federal standards resulted in software that was safer or more cost-effective. In particular, with NIST's efforts to govern the software development lifecycle, the scope grew so broad that it became infeasible to measure compliance. For example, NIST sought to standardize the entire process of software documentation, across all possible software projects. Because software projects can vary immensely, the so-called standard boiled down to highly subjective decisions on a project-by-project basis. Even those who tried to comply with the standard were unsure whether it led to better documentation, let alone better performance. Similarly, NIST took on the task of standardizing the entirety of software validation and verification, an even more daunting exercise. Although NIST could enumerate different categories of tests, it was unable to recommend anything more concrete than that each vendor should create and maintain its own testing plan.

Third, NIST was unable to locate expert consensus on best practices for those higher-level activities. Consequently, NIST relaxed the criteria for compliance, which further diminished the utility of conforming to those standards. The value of standardization lies in achieving a desired uniform attribute and in streamlining the decision-making process to get there. Instead, NIST encouraged software practitioners to use their own judgment on core aspects of the software development process. FIPS compliance became an exercise in creating extra paperwork to justify one-off design decisions, rather than facilitating and channeling those design decisions into a consistent mold. In other words, NIST's software standards added substantial costs for dubious benefits.

To be clear, NIST's work was a best efforts campaign, and its shortcomings likely reflect the deeply challenging nature of the mission, rather than errors in execution. It is thus dismaying to see NIST being tasked again with even broader missions.

Self-Governance Frameworks

In its latest return to software standards, NIST has doubled down on the malleable, broad-church approach. The risk management frameworks across cybersecurity, secure software development, and AI all share striking similarities in asking vendors to engage in self-study of risks and to self-select appropriate precautions and remediations. Because wrong answers are vanishingly rare, any entity can call itself an adopter without making real changes to its software practices.[149] What NIST's latest frameworks offer at most is a common lexicon that amalgamates many different practices and schools of thought. Not surprisingly, this model of voluntary compliance has drawn fire for failing to generate effective results.[150]

NIST's risk management frameworks cannot be used to determine an objective standard of care, because they do not dictate any particular set of conduct. For example, there is no measurable equivalence among entities that call themselves "Tier 1" or any other tier. Moreover, the tier numbers have no correlation with actual likelihood of failure or exploitation. In many or most cases, adopting such a framework merely means the entity has generated documentation to justify the practices it already performs. Credulity is strained, therefore, when lawmakers purport to use compliance with a NIST framework as per se proof of reasonable care.[151] Perhaps one could argue instead that *failure* to follow a NIST framework falls below the minimum threshold of reasonable precaution. But even this argument is tenuous, since there is no clear evidence that compliance leads to improved outcomes.

A fair question to ask is why NIST, whose core competency is in standard-setting, would promote voluntary, self-governance frameworks over conventional technical standards. An optimistic argument is that NIST's big-tent strategy offers a gentle on-ramp that builds toward a culture of compliance. In other words, even if a risk management framework does not offer uniform standardization today, it encourages collective buy-in that can then be used to exert upward pressure tomorrow. Once there is a critical mass of participation, it becomes difficult for entities to exit the framework. While the

---

[149] Although the risk regulation approach can be criticized for addressing only "measurable, quantifiable harms" (see Kaminski, supra note 136, at 14), the lack of quantifiable metrics is equally if not more concerning.

[150] See, e.g., Melanie Teplinsky, "A Review of NIST's Draft Cybersecurity Framework 2.0," *Lawfare*, Sept. 13, 2023 (stating that "voluntary compliance with the framework has largely failed to generate effective cybersecurity" and that the updated framework (CSF 2.0) "is unlikely to fundamentally improve the nation's cyber posture").

[151] See supra note 5 and accompanying text.

participation theory is compelling in many aspects, participation alone is an empty prize unless there are meaningful criteria for exclusion. At least in the software context, it seems unlikely that there are new, objective metrics on the horizon that NIST is waiting to spring on participants.

A more pessimistic response is that NIST has been assigned a mission it knows it cannot fulfill. Whereas the Brooks Act of 1965 offered the agency greater leeway to define its own mission, the recent executive orders have tasked NIST with pointed charges to improve cybersecurity performance and AI performance as a whole. Those orders create unrealistic pressure to provide rapid solutions to problems that NIST has studied for many decades without great success. The fact that NIST has shied away from technical mandates is a telling signal that the root problem of software quality is something more nuanced than mere lack of coordination by a central authority.[152] NIST's extensive experience with the FIPS process also shows that ponderous, multi-year standard-setting processes have been ineffective at shaping software practices. By contrast, a nonsubstantive framework can be released quickly and allows NIST to show it is taking action. Just as important, a self-governance framework allows NIST to pass through the intractable aspects to third parties. If this explanation is plausible, the best-case outcome for NIST's frameworks is if they can motivate—or even compel—more consistent, rigorous information-gathering across a fuller range of software practices.[153]

Institutional Competence

Ultimately, NIST's experience should teach us that the centralized agency model is not a silver bullet. Typically, the comparative advantage of federal agencies over other institutions is that agencies are more competent at assembling expert knowledge, establishing uniform rules based on that expertise, and efficiently policing those rules on a national basis.[154] Especially for matters involving informational asymmetries or collective action problems, a central regulator may be better able to facilitate a policy outcome than courts, legislatures, or private market forces. Accordingly, many commentators have championed the central agency model as a quick fix for establishing a new software standard of care.

Yet, when experts in the field lack knowledge or consensus about a policy issue, the comparative advantage of agencies is correspondingly diminished. Unable to draw upon that collective expertise, the agency will likely struggle to provide uniformity and efficiency as well. Tellingly, software experts have long agreed that there is no easy way to measure or certify software quality. Some believe Microsoft's

---

[152] For example, Cary Coglianese has argued that performance standards do not work well when the regulated entities are highly heterogeneous and the regulators lack capacity to measure outputs or outcomes. See Coglianese, supra note 119, at 546–47 & fig. 1.

[153] See Dempsey, supra note 119 (calling for the establishment of a Bureau of Cyber Statistics that would gather information about cyber incidents in order to facilitate the ability to quantify risk and to measure outcomes).

[154] See Choi, "Institutional Choice," supra note 8.

model of secure software development represents a high-water mark. Others feel strongly that conservative methods that hew closer to waterfall engineering processes are the only way to assure quality in safety-critical contexts. Still others argue to the contrary that Agile methods are both necessary and safer because they are more nimble.[155] In the end, most companies adopt their own bespoke approach. Disagreement persists because none of these methods can be evaluated with any confidence, and all continue to produce faulty software with unknowable rates of error.

Given the deep-seated uncertainty among the software expert community, NIST is unlikely to forge a new, independent path forward, especially because it is a nonpartisan technocratic agency with a strong institutional culture committed to scientific knowledge. That commitment means NIST is less likely to adopt policy positions that are not grounded in expert consensus. Doing so would threaten NIST's internal culture and institutional reputation as a neutral arbiter of technological standards.

NIST is also unusual in that it is a nonregulatory agency, meaning that it cannot directly enforce its own rules. Although federal procurement policies from the Office of Management and Budget often require compliance with NIST standards, moving the larger commercial market depends on buy-in from area experts and the public at large. Even within the federal government, a major mismatch with the commercial market makes internal compliance challenging to achieve—studies have repeatedly shown for decades that federal agencies remain badly noncompliant with NIST's software-related standards.

A new or different agency could take a more aggressive stance,[156] but it is questionable how different the outcome would be. One model is the Department of Defense, which sought for many years to exert rigid control over its software standards. In the end, the cost-benefit ratio proved too high to be sustainable for all but the most critical software applications. A second model is the Food and Drug Administration (FDA), which deals with software only in the limited context of medical devices. It is possible that a narrower mission could make it easier to define clearer metrics in specific domains. Even so, the FDA has struggled mightily to define such rules.[157] A third model is the Federal Trade Commission, which has sought to bring case-by-case enforcement actions against entities for failure to maintain reasonable software practices. Greater scrutiny and contestation of software practices is much needed. That said, when the agency is unable to provide *ex ante* rules or guidance, then case-by-case

[155] For a brief discussion, see Choi, "Software as a Profession," supra note 8, at 578–79.

[156] See Matthew T. Wansley, "Regulation of Emerging Risks," *Vanderbilt Law Review* 69 (2016): 401, 431 (arguing that a central agency should be empowered "to implement a moratorium if it could demonstrate that an emerging technology *plausibly* created a significant risk to health, safety, or the environment," because it would "allow agencies to act notwithstanding scientific uncertainty").

[157] See Nathan Cortez, "Regulating Disruptive Innovation," *Berkeley Technology Law Journal* 29 (2014): 175.

adjudication does not offer much comparative advantage over the judiciary,[158] and such actions raise vagueness and due process concerns.[159]

Finally, it is conceivable that the AI field is meaningfully different, and that the centralized agency model will be more effective at standardizing AI practices than it has been at standardizing software practices.[160] Although NIST has studied AI for many years, it has only just begun its efforts to standardize AI, so there is less historical evidence to predict how that task will compare.[161] For example, like cryptography, the core AI methods are highly mathematical and thus more conducive to quantitative metrics. NIST's ability to certify standardized data sets offers tremendous upside. For the moment, the number of qualified AI practitioners remains relatively small, and the range of techniques with real-world impact remains relatively narrow, which could facilitate convergence toward a consensus code of conduct. To be most effective, NIST must find a way to compile, catalog, and publish information about actual AI modeling practices at the handful of major entities that are leading the field. It should avoid generalities and idealizations that do not match commercial realities. And it must find ways to show a clear payoff for practitioners to adopt NIST's standards and guidelines.

CONCLUSION

As software-related harms continue to burgeon into public crises, the federal government has placed its trust in NIST to conjure up new performance standards and accountability measures. In response, NIST has issued a series of voluntary, self-governance frameworks, rather than traditional, uniform standards. Those weak frameworks trace back to older work by NIST, which vigorously tried to standardize a

[158] Cf. *Kisor v. Wilkie*, 139 S. Ct. 2400, 2417 (2019) ("When the agency has no comparative expertise in resolving a regulatory ambiguity, Congress presumably would not grant it that authority.").

[159] See *LabMD, Inc. v. FTC*, 894 F.3d 1221, 1236 (11th Cir. 2018) (vacating the FTC's cease-and-desist order as unenforceable, because it "is devoid of any meaningful standard informing the court of what constitutes a 'reasonably designed' data-security program"); Justin (Gus) Hurwitz, "Data Security and the FTC's UnCommon Law," *Iowa Law Review* 101 (2016): 955; cf. *West Virginia v. EPA*, 142 S. Ct. 2587, 2612 (2022) (disallowing an agency's authority to effect a "fundamental revision of the statute" where there is "little reason to think Congress assigned such decisions to the Agency"); Tschider, supra note 4, at 118 n.151 (observing that administrative adjudicative approaches have "reached substantial roadblocks").

[160] See Bryan H. Choi, "AI Malpractice," *DePaul Law Review* 73 (2024): 301.

[161] See NIST, "Artificial Intelligence Measurement and Evaluation at the National Institute of Standards and Technology" (2021), at 1–2, https://perma.cc/9QF8-K2SA (explaining and cataloging NIST's long history of measuring and evaluating AI technologies in areas including information retrieval, speech and language processing, computer vision, biometrics, and robotics, but noting the need to look at properties "beyond performance accuracy measurement that were historically viewed as outside the purview of AI"); see also NIST, "Artificial Intelligence Measurement and Evaluation Workshop Summary" (2021), https://perma.cc/ZKL5-DRJL.

broad range of software practices. NIST failed for a number of reasons, including the slow pace of the standard-setting process, the breadth and complexity of the standard-setting scope it set, and—most importantly—the lack of expert consensus on best practices.

It is tempting to be lulled by NIST's risk management frameworks into false illusions of compliance and safety, even if the frameworks are not intended to be used in that manner.[162] A close examination of those frameworks reveals that there is little new substance on offer. They do not impose any measurable outcomes, performance metrics, or other meaningful guardrails. NIST's frameworks cannot be used as a quick shortcut for the vexing problem of defining a software liability rule.

In the end, the moral of NIST's story may be that there cannot be a single "reasonable" standard of care for software liability. After decades of study, NIST has embraced a heterodoxic model of professional self-governance that eschews any single methodology or school of thought.[163] Other standard-setting bodies that have studied the issue have reached strikingly similar conclusions. Perennial calls to appoint a centralized authority to promulgate new software standards should look first to the vast body of work that NIST has already produced, before expecting different outcomes.

The Digital Social Contract paper series is supported by funding from the John S. and James L. Knight Foundation and Meta, which played no role in the selection of the specific topics or authors and which played no editorial role in the individual papers.

[162] See, e.g., NIST, Cybersecurity Framework, supra note 113, at 2 (explaining that "phrases like 'compliance with the Framework' can be confusing and can mean something very different to various stakeholders").

[163] Cf. Choi, "Software as a Profession," supra note 8.