



“You have to read 50 different RFCs that contradict each other”: An Interview Study on the Experiences of Implementing Cryptographic Standards

Nicolas Huaman and Jacques Suray, *Leibniz University Hannover*; Jan H. Klemmer, *CISPA Helmholtz Center for Information Security*; Marcel Fourné, *Paderborn University*; Sabrina Klivan, *CISPA Helmholtz Center for Information Security*; Ivana Trummová, *Czech Technical University in Prague*; Yasemin Acar, *Paderborn University & The George Washington University*; Sascha Fahl, *CISPA Helmholtz Center for Information Security*

<https://www.usenix.org/conference/usenixsecurity24/presentation/huaman>

**This paper is included in the Proceedings of the
33rd USENIX Security Symposium.**

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

**Open access to the Proceedings of the
33rd USENIX Security Symposium
is sponsored by USENIX.**

“You have to read 50 different RFCs that contradict each other”: An Interview Study on the Experiences of Implementing Cryptographic Standards


Nicolas Huaman ^L

Jacques Suray ^L


Jan H. Klemmer ^C

Marcel Fourné ^P

Sabrina Klivan ^C

Ivana Trummová ^T

Yasemin Acar ^{P,G}

Sascha Fahl ^C

^L Leibniz University Hannover

^P Paderborn University

^T Czech Technical University in Prague

^G The George Washington University

^C CISA Helmholtz Center for Information Security

Abstract

Implementing cryptographic standards is a critical process for the cryptographic ecosystem. Cryptographic standards aim to support developers and engineers in implementing cryptographic primitives and protocols. However, past security incidents suggest that implementing cryptographic standards can be challenging and might jeopardize software and hardware security. We need to understand and mitigate the pain points of those implementing cryptographic standards to support them better.

To shed light on the challenges and obstacles of implementing cryptographic standards, we conducted 20 semi-structured interviews with experienced cryptographers and cryptographic software engineers. We identify common practices when implementing standards, including the criticality of reference and third-party implementations, test vectors to verify implementations, and the open standard community as central support for questions and reviews of implementations.

Based on our findings, we recommend transparent standardization processes, strong (ideally formal) verification, improved support for comparing implementations, and covering updates and error handling in the standardization process.

1 Introduction

Cryptography is one of the pillars of modern information security. Many software and hardware products implement cryptographic primitives, algorithms, and protocols to protect data in transit and at rest. For example, Transport Layer Security (TLS) [65] is used to protect the vast majority of network traffic on the Internet [17] and end-to-end encryption in messaging apps such as WhatsApp [41] or Signal [68] protects billions of messages and users every day. However, as demonstrated by previous work, deploying and using these important cryptographic standards can be a difficult and laborious process [40, 61, 24, 73, 16].

A critical and particularly challenging step in the deployment path is the implementation of cryptographic standards.

Standardization organizations such as the Internet Engineering Task Force (IETF) [30] and the U.S. National Institute of Standards and Technology (NIST) [49] publish standard documents that aim to serve as a blueprint for implementations that are secure and provide inter-compatibility. Cryptographic standards often include details of primitives and protocols, mathematical information for secure implementations, references to research papers, and other documents meant to support the implementation of primitives, algorithms, and protocols. In other cases, details are missing or vague, resulting in gaps in the standard that increase the risk of implementation mistakes and require extensive cryptographic knowledge. Incidents illustrate potential consequences:

MITRE’s CWE-1240 list of common weakness enumerations (CWE) provides an overview of risky implementations of cryptographic primitives [33]. An example is the widely used OpenSSL library, which included a vulnerable implementation of the ChaCha cipher, used in TLS 1.2 onwards (see Request for Comment (RFC) 7905 [36]). Even though RFC 7539 [50] includes a specific byte size interval, the implementation allowed using a byte size outside the interval. This deviation from the standard resulted in a vulnerability that enabled attacks on confidentiality and integrity [66].

Similar to standard documents, reference implementations, and other third-party implementations are critical pieces in cryptographic standardization processes. For example, the IETF provides the Secure Shell (SSH) protocol [42], which is used in the OpenSSH implementation [62]. As this implementation is commonly used as a reference and provides options not included in the SSH protocol [42], it is a *de facto* extension to the standard that is not governed by the IETF. The above examples illustrate some challenges when implementing cryptographic standards.

While previous work investigated end-user challenges using cryptography [35, 71, 5], developers struggling with the correct use of cryptographic APIs [19, 31, 1, 16], and the use of cryptography in industry contexts [21], an in-depth understanding of the experiences and challenges when implementing cryptographic standards is missing in the literature.

Previous research on cryptographic standards focused on vulnerabilities in standards or faulty implementations [74, 12, 15, 61, 24], but did not consider the implementation process itself. We present and discuss in-depth insights from 20 semi-structured interviews with experienced implementers of cryptographic standards. We illustrate the challenges they encounter from the perspective of new and veteran developers in the cryptographic community and from an industry, academic, and hobbyist view. We hope our findings can contribute to easier-to-implement future cryptographic standards.

Research Questions. In this work, we address the following research questions:

RQ1: *Which are developers' practices when implementing cryptographic standards?* Implementing cryptographic standards is essential to deploy cryptography from research into products. We are interested in better understanding developers' implementation practices and requirements for cryptographic standards.

RQ2: *What are the challenges of implementing cryptographic standards?* Cryptographic standards are complex and require a high level of expertise to understand and follow. We aim to understand the challenges of writing cryptographic software following standards.

RQ3: *How can future cryptographic standards be improved to better support implementations?* The correct and secure implementation of cryptographic standards is crucial for overall software security. We aim to understand better how future cryptographic standards can be improved to support cryptographic software developers and implementations.

We interviewed 20 implementers of cryptographic standards. Most of them had more than ten years of experience and worked in academia or industry on application areas of cryptography ranging from PGP, TLS, elliptic curve cryptography (ECC), and post-quantum cryptography (PQC). All had critical roles in their projects. Participants illustrated the criticality of reference and third-party implementations, formal verification, and test vectors when implementing cryptographic standards. They reported that managing standard updates, multiple variations and complexity of standards, error and typo handling, and patents are major challenges during implementation. Participants expressed a strong preference for transparent and open standardization processes and a need for improved communication in the community.

To support the transparency and replicability of our work, we make all non-identifying research artifacts available, including our semi-structured interview guide, the codebook, and the invitation emails in the [Availability](#) section. Furthermore, we provide a background on cryptographic standardization processes in [Appendix A](#).

2 Related Work

We discuss related work in two key areas: the security of cryptographic standards, and challenges regarding adoption and implementation of cryptography.

Security of Cryptographic Standards. Cryptographic software is commonly based on public standards, typically provided by standardization bodies such as NIST or IETF. Hence, the quality of cryptographic software is strongly connected to the quality of the cryptographic standards they implement. Due to this, previous work on standards' security often focuses on the quality of the cryptographic standardization process and standard candidates. Kannwischer et al. examined submissions to the official NIST PQC standard effort, and found that almost all the required implementations were vulnerable despite passing all stages of the standardization process [32]. Similarly, Mouha et al. discovered a vulnerable standard implementation that reached the final round of the NIST SHA-3 competition in 2011 [48]. Hao evaluated the IETF standard selection process and discussed principles to improve future security standardization processes [22].

Security flaws can also be present in established standards. For example, Woodage et al. analyzed the NIST SP 800-90A standard. After finding some security flaws, they stress the criticality of the underlying algorithms and that algorithms should not be standardized until their security was proven [74]. Do et al. found vulnerabilities in the official W3C Web Payments API [12]. Feng et al. detected logical contradictions in several CoAP RFCs on a statement level. Here, quality problems did not arise within the implementation, but the description scope [15]. Other affected standards include the PGP protocol. Halpin analyzed it, investigating and demonstrating the central design flaws that make it difficult to adapt PGP to modern requirements or to deprecate older, less secure versions [20]. To remediate these issues, Almeida et al. proposed a framework for automatically generating functionally correct reference implementations from cryptographic standard specifications [2]. Mohajerani et al. presented a framework to evaluate the side-channel resistance of Lightweight Cryptography for the Standardization Process [46].

While related work often focuses on the theoretical issues of cryptographic standards, our interview study investigates the practical experiences with and perceptions of standards from cryptographic software implementers. We aim to understand the impact of cryptographic standards on the security of cryptographic software to recommend improving future cryptographic standards and standardization.

Cryptographic Adoption & Implementation Challenges. When standards lack necessary support and information, developers can face difficulties implementing these standards. These adoption issues and overall cryptographic misuse can lead to severe security issues. For example, Li et al. found that two thirds of Apple iOS apps contained security flaws caused by cryptography misuse [40]. Similarly, Oltrogge et al.

scanned 1.3 million apps from Google Play, identifying over a thousand candidates with vulnerable certificate validations. Despite providing secure defaults, standard definitions stated optional or adjustable parameters that provoked implementation vulnerabilities due to inadequate documentation and developer usability [61]. Hebrok et al. found vulnerable implementations of the optional non-standard TLS session ticket mechanism. Respective implementers could not construct secure keys and assess potential risks of reusing keys [24]. After investigating vulnerabilities in TLS-DH(E), Merget et al. reached similar conclusions and found that secret construction and forward secrecy are not sufficiently secure [45]. Valenta et al. found in an internet-wide scan that ECC implementations often deviated from the respective standard definitions. Among other problems, these implementations lacked validity checks given by the standards [73].

A systematic review by Lazar et al. examined 269 entries on software vulnerabilities from the CVE database. They found that 17% of the issues were located in cryptographic libraries [38]. To discover vulnerabilities in implementing security measures, Rahaman et al. presented rules for source code analysis that effectively perform vulnerability checks at compile time [64]. Gorski et al. explored the use of cryptographic APIs. To create recommendations to increase security, a focus group study ($n = 25$) explored feedback to developers and review what types of security information are helpful [19]. Another study by Jancar et al. invited 44 developers of cryptographic libraries to participate in a survey and found differences between academic knowledge and cryptographic engineering. Finally, they provide recommendations on how developers can prevent timing attacks [31]. Close to this work, Haney et al. interviewed developers to gain insights on using cryptographic measures. They suggest assessing the usability of standard documents regarding the resulting cryptographic quality of real-world implementations [21]. Finally, Fischer et al. presented an interview study with cryptographic experts regarding the application of cryptographic research. They identified key challenges in adapting the results of cryptographic research, notably identifying conflicting goals between the different stakeholders of the cryptographic ecosystem [16].

Adding to this body of research, we contribute insights on the interactions between standards and standardization bodies, and the developers implementing them. We are specifically interested in what makes a good cryptographic standard for cryptographic software developers. We further examine the obstacles these developers face with the ecosystem and resources around cryptographic standards. As a result, we identify core challenges they face and provide a set of recommendations for writing standards, reference implementations and other resources.

3 Methodology

Using semi-structured interviews, we collect insights into various workflows from developers who directly interact with, read, and implement cryptographic standards. In this section, we illustrate our study’s design and recruitment process. We provide an overview of our methodology in [Table 1](#).

Table 1: Overview of our methodology, including interview guide design, data collection, and analysis.

Stage	Additional Information
1. Interview Guide	
<i>Interview Design</i>	Feedback round with multiple experts
<i>Piloting</i>	2 pilots interviews
<i>Recruitment</i>	Projects, mailing lists, snowball sampling
2. Interviews	
<i>Interviewers</i>	1 (+1 shadow) interviewer
<i>Participants</i>	20, avg. 78 minutes
<i>Structure</i>	S1: Projects and Demographics S2: Process of Implementing Standards S3: Challenges of Implementing Standards S4: Perception of Cryptographic Standards S5: Problems of Current and Future Standards
3. Data Analysis	
<i>Transcripts</i>	Amberscript [4]
<i>Methodology</i>	Thematic Analysis [10]
<i>Coding</i>	5 coders, 2 per transcript

3.1 Interview Design

To draft an initial interview guide, all authors collaboratively discussed relevant interview topics, obtaining a few high-level leading questions to ask participants. We supplied each of those questions with more profound, conditional sub-questions to collect in-depth insights during the interviews and prepare for various topics that might emerge during interviews. During and after both of these phases, we obtained additional feedback from co-workers and students and internally tested the interview guide to arrive at a final internal draft.

In the next step, the interview guide draft was reviewed by different experts that were either cryptographic practitioners or experienced with qualitative research. We iteratively improved the interview guide based on the experts’ feedback.

3.2 Interview Guide

After completion of our guide, we arrived at five general topics, described below. The complete interview guide is available in our replication package (cf. [Availability](#) section).

Intro and Consent. We start with a brief introduction of ourselves, our research field and the interview motivation. We explained that we do not judge any answers, that all questions can be skipped, and that the interview can be ended at

any time. Furthermore, we explained how we process and anonymize the interview answers to protect participants' identity. Finally, we asked participants for their consent to an audio recording of the interview, and gave them the opportunity to drop out and have their data deleted at any time.

S1: Projects and Demographics. We asked general questions about participants' experiences, projects, and standards to get started. For example, we asked about their background and education in cryptography, and the purpose of their projects. We also asked how they got started working with standards.

S2: Processes of Implementing Standards. Next, we focused on the considered resources and processes. We also asked about internal and external testing, and reviewing of implementations.

S3: Challenges of Implementing Standards. In this section, we asked participants about challenges, including legislation and updates to standards.

S4: Perception of Cryptographic Standards. This interview section covered the experience and perception of standard documents and bodies. We asked participants about specific standards and organizations they worked with in the past, and what makes a good standard document for them.

S5: Problems of Current and Future Standards. Finally, we cover the wishes and needs of participants regarding current and future standards. We presented an example scenario surrounding legislation and their opinion on that, and we asked about any remaining problems they encountered with standards. Finally, we asked them about their trust in current implementations and their wishes for standards in general.

Outro/Debriefing. After the interview, we asked them if they had any additional remarks, thanked them, and stopped recording. To conclude, we asked for potential followup candidates they knew as part of our snowball sampling.

3.3 Recruitment

Due to the small population of experienced cryptographic practitioners who qualified for our study, recruiting suitable participants was challenging. To find potential participants, we identified open-source projects implementing cryptography. A project was included if (i) it implements a cryptographic function, and (ii) if the last commit was not older than two years. We examined the top 120 viable projects in the GitHub topic *cryptography*, sorted by stars. Furthermore, we investigated dependencies of projects from the cryptographic area, which led us to find and include projects outside GitHub, including OpenSSL¹ and NaCL². We then identified and contacted the developers who were among the most active in implementing cryptographic functions. To avoid issues with

¹<https://www.openssl.org/> (Accessed 2024-06-14)

²<https://nacl.cr.yp.to/> (Accessed 2024-06-14)

GitHub's terms of service,³ we only contacted developers who provided public contact channels external to GitHub, e.g., via personal or professional websites. To identify additional matching projects, we also used the *awesome-cryptography* list [70]. After each interview, we used snowball sampling by asking study participants if they could recommend other potential participants. Finally, as some interviewees suggested, we sent out interview invitations to two major mailing lists regarding cryptographic standardization, namely the NIST pqc-forum [60] and the IETF SAAG mailing lists [29]. While the first is used for NIST's PQC standardization effort, the latter reflects the Security Area Open Meeting and further addresses participants of the IETF 118 Prague meeting [27] that took place at the time of recruitment. This way, we reached cryptography stakeholders outside the open-source development space.

3.4 Qualitative Data Analysis

We audio-recorded all interviews and transcribed them using the GDPR-compliant transcription service *Amberscript* [4]. We manually reviewed all transcripts for completeness and correctness. We performed *thematic analysis* [10] to analyze the transcripts for common themes and topics. We first coded the interview to derive general opinions and experiences. Then, we used an iterative process to refine our codes into generally applicable themes and topics, which were then fit to the individual citations in each interview.

Two researchers coded each interview independently. After individual coding, both researchers merged and discussed their coding, agreeing on codes, and refining the codebook. This was done for all 20 interviews, ensuring that the resulting codebook captures the content of all interviews. As every merging discussion ended with consensus (theoretical agreement of 100%), we do not calculate inter-rater reliability (IRR) [44]. Using the resulting codebook, we collectively derived themes in multiple iterative sessions with four coders. We provide the codebook as part of our replication package (see [Availability](#) section).

3.5 Ethics

Our institution's ethical review board (ERB)⁴ approved this interview study. This work follows the ethical principles of the *Menlo Report* for research involving information and communications technologies [34]. Additionally, we adhered to the strict German and European data and privacy protection laws, including the General Data Protection Regulation (GDPR). We audio-recorded all interviews, used a GDPR-compliant transcription service, and deleted audio recordings

³<https://docs.github.com/en/site-policy/acceptable-use-policies/github-disrupting-the-experience-of-other-users> (Accessed 2024-06-04)

⁴Equivalent to an IRB.

after their successful transcription. After the interview, we stored transcripts in a self-hosted, secure cloud collaboration software, encrypted all data at rest and in transit, and only granted researchers involved in the project access to the data. We offered participants compensation in the form of a \$60 donation to an open-source project of their choice. We informed participants about these conditions via a consent form that they had to accept in the pre-survey, which was included in the informational material provided for recruitment. They could further discuss and ask questions about the form during the introduction of our interview⁵. As interview questions might be sensitive, we informed participants they could skip questions or terminate the interview at any time. Finally, we provide interviewees with a preprint to suggest changes or veto their contribution to the camera-ready version. To protect participants' privacy, we removed identifying information (e.g., names, companies, project information) from any quotes in this paper.

3.6 Limitations

This work has several limitations inherent to qualitative research and interview studies, including limited generalizability and biases such as recall and social desirability biases [39]. We accounted for these biases by interviewing a diverse sample of experts who fit our recruitment criteria. Most importantly, we focused on interviewees who had previously implemented cryptographic standards.

We recruited some interviewees by purposive sampling, using our personal and professional contacts. Hence, most interviewees have a similar background and are primarily from the US and the EU. We did not obtain a balanced or diverse sample of participants regarding personal recruitment criteria like gender, ethnicity, and age. This means our sample is unfit to provide meaningful statements about underrepresented parts of the cryptographic community or the effect these biases may have on cryptographic discourse. However, we recruited interviewees using snowball sampling and through cryptographic mailing lists and forums such as the NIST pqc-forum and the IETF SAAG mailing list. This was done to obtain a diverse sample regarding years of experience, work context, and project areas for our interview within the relatively challenging small group of cryptographic developers working with standards. We developed our interview guide with these limitations in mind.

To reduce social-desirability bias, we specifically mentioned that we were only interested in our participants' experiences and opinions and did not judge them in any way. We also clarified that interviewees could skip questions or end the interview for any reason. As in any self-reporting user study, interviewees' over- or under-reporting experiences and opinions could bias our results.

⁵The consent form is part of our replication package (see [Availability](#) section).

4 Results

In the following, we outline results from our interviews with 20 experienced implementers of cryptographic standards. The interviews were conducted in English, except for three conducted in German based on participant preferences; quotes from these are translated in the paper.

4.1 Interviewee Demographics

Due to the small sample size and well-connected cryptographic community, we provide only aggregated demographics in [Table 2](#) to protect the pseudonymity and privacy of our participants.

Table 2: Demographics of our 20 participants.

Personal		Project-related	
Years of Experience		Project Roles	
Less than 5 years	4	Developer: Lead	9
5 to 20 years	9	Developer: Team member	3
More than 20 years	7	Developer: Co-leader	3
Work Context*		Developer: Sole contributor	2
Academia	10	Other	3
Industry	10	Project Areas*	
Hobby	7	Post-Quantum Cryptography	15
Open source dev	5	Elliptic Curve Cryptography	6
Crypto side project	3	TLS, SSL	5
Education*		PGP	5
Academic	15	Other Crypto Libraries	4
Self-taught	11	Formal Verification	3
On the job	7	SSH	1
Training course	1	Project Open or Closed Source*	
Gender		Open source	18
Male	8	Closed source	2
Female	1	Other/NA	1
Non-binary	1	Recruitment Channel†	
Not Shared	10	Projects	7
		Snowball Sampling	7
		Mailing List	6

* Multiple answers possible, may not sum to 100%.

† Participants may fit multiple channels, best match reported.

NA = not applicable

Overall, our participants tended to be highly experienced. Only four participants reported having worked in cryptography for less than five years, while 16 had at least five years of experience, and seven participants reported more than 20 years of experience in cryptography.

Our participants filled various roles in their projects, often more than one, but mainly worked on implementing cryptography. Among them, nine had a leading role, while three considered themselves as co-leaders, both part of a larger team. On the other hand, two participants reported being their project's main or sole contributors and only receiving minor external help, e.g., occasional pull requests from other developers. Besides the majority of developers, three participants mentioned different roles, including teaching and consulting

activities, where they mainly work on prototypes or other non-production-ready code.

Participants also mentioned multiple sources regarding their education and training in cryptography. Most commonly, 15 participants reported academic education and university courses; eleven mentioned being self-taught. Less common, seven stated to have gathered cryptographic knowledge on the job. Only one participant mentioned formal training. Ten participants worked with cryptography in an academic context, e.g., as a part of research or university courses. The remaining ten worked in industry. Additionally, some worked on cryptography projects in their spare time, e.g., seven reported hobby projects, and five mentioned open-source projects. Fifteen participants were mainly involved in post-quantum, mirroring current trends in the cryptographic community. However, some also worked on, e.g., ECC, TLS, and SSL, PGP, or other cryptographic libraries.

Eighteen participants reported working on an open-source project, while two were part of closed-source projects. Finally, one participant did not actively contribute to development, as they mostly focused on teaching cryptography to students.

Furthermore, we report distribution across our recruitment channels. As multiple may apply, for example, we might have directly contacted somebody who also saw us on a mailing list. We chose the channel that best applied: *Snowball sampling* in case participants recommended or contacted other participants; *mailing list* in case participants responded to our mailing list post; *projects* for all other participants who responded to our personal invite email. We recruited 7 participants based on our list of GitHub projects. Furthermore, 6 participants signed up after we promoted our study via the NIST pqc-forum and IETF SAAG mailing lists. We also asked participants for potential followup contacts. In 7 cases, the recruitment snowballed as participants explicitly recommended a person to us. This led to industry contacts and participants with little cryptographic experience, broadening our sample.

4.2 Practices for Implementing Cryptographic Standards

In the second interview section, we asked participants about their preferred resources for implementing a cryptographic standard, their implementation process, and how they test their code or handle issues like gaps in the standards. Gaps, as we refer to them, are any missing details in a standard needed for the implementation, e.g., due to implicit assumptions or lacking detailed explanations in standards. When encountering gaps, developers must decide themselves or consult other implementations to find out how to close them, which can lead to vulnerabilities or mistakes. The following discusses the reported topics and practices for the cryptographic standards implementation process. Starting with this section, we replace the participant counts with tendencies and qualifiers

to protect their identities.

Implementers Appreciate Reference Implementations. Regarding resources, almost all participants mentioned using reference or third-party implementations as their main resource for implementing cryptographic standards. As one participant stated:

“Basically when software is written in a reasonable language for cryptography, then it’s unambiguously specifying, ‘Here’s what you’re supposed to do.’ That’s a very important resource in reference implementations that are clearly specifying something. Sometimes the simplest reference implementations are in C rather than Python. [...] A good example is the MD5 specification. The original MD5 specification was just, ‘Here is software that computes MD5.’” — P3

This illustrates the appreciation of reference implementations as a blueprint for standard-conforming implementations. Participants mentioned relying on reference implementations to close gaps in standard documents, obtaining suggestions for implementation details not specified in the standard, conducting conformance testing, and generating test cases for test suites. Almost all participants reported using standard documents in some form, preferably with additional resources, such as scientific papers and reference implementation.

“I do like standards that are written in a way that you can match them up to. Here’s the software that’s supposed to be matching the standards. Here are the papers that are supposed to be matching the standards.” — P3

More in-depth, some participants reported comprehension challenges of standard documents due to their technical complexity, length, outdated or burdensome format, and language. They mentioned reference implementations as critical information sources to help comprehend cryptographic standards.

Formal Verification Increases Trust in Standards and Helps with Testing. Another topic that our participants mentioned is the criticality of formal verification and mechanized proofs of standards and their implementations. Some participants reported involving some form of formal verification to confirm certain security properties in their cryptographic implementations. A few participants further mentioned using formally verified implementations to generate test vectors. This illustrates the benefits of mechanically checked formal correctness proofs.

About half participants described formal verification as a criterion of quality for standards and would appreciate formal verification to be required for future standards. While formal verification for cryptographic standards is costly and laborious and might be an obstacle to submission, it offers significant advantages, as one participant puts it: *“I think the highest trust*

I would now have is in these high-assurance implementations that have been formally proven. [...] Even if it's not Kyber, it is secure with this proof.” (P17)

Standards Should Provide Test Vectors. While not offering the same security guarantees as formal verification, most participants discussed the criticality of test vectors. Test vectors are known-answer tests that associate an input value with an expected outcome, e.g., a key and some plaintext as input and ciphertext as expected output. Test vectors allow for verification of the correctness of an implementation for the given test vectors. They also support *test-driven development*, as a few participants noted. About half of the participants mentioned using test vectors to test their implementations or expect them as part of a high-quality standard. As one participant summarized it:

“Test vectors where possible. Even test vectors that test like negative stuff, things that should not work or behavior, let's say, outside. I don't know: ‘You should not validate this point. It should not work with your implementation.’” — P14

Generally, our participants distinguished two sources of test vectors: (i) official test vectors provided by the standard and (ii) unofficial test vectors from trustworthy third parties. While many participants discussed the benefits of test vectors, not all standards provide them: *“Test vectors would be very helpful. Test vectors everywhere, but that is not the case. KMAC, for example, has no test vectors.”* (P18). Especially in these cases, test vectors from trustworthy third parties are critical. For example, a participant mentioned frameworks like *Project Wycheproof* [18] as a trustworthy source for test vectors.

Trustworthy Third-Party Implementations Provide Additional Support. In addition to official reference implementations and test vectors, our participants further discussed the benefits of trustworthy third-party implementations. For example, they reported to have used large-scale open-source libraries such as *OpenSSL* or *OpenSSH* for the TLS and SSH standards, as test systems or blueprints for their implementations. One participant mentioned this as an integral part of their implementation process: *“The more implementations the better. We use a mixture of open-source implementations, proprietary implementations from other vendors, and then our implementations.”* (P13).

About half of our participants mentioned using third-party implementations with widespread adoption to help ensure new implementations' compatibility with existing ones—especially popular and widely used ones. Testing interaction and compatibility with other implementations were part of about half of participants' workflow. One participant mentioned that having multiple implementations of a standard is a sensible requirement:

“In principle, it is an IETF requirement that there must be at least two implementations that can in-

teroperate. [...] this is certainly a sensible requirement in practice. [...] Because the more implementations there are, the more errors we find.” — P15

The above IETF requirement originates from the *IETF Standards Process* [8, 11] and aims to establish the availability of implementations for cross-compatibility testing. Additionally, participants mentioned that third-party implementations provided them a reference for gaps or undefined details, and help them implement the standard correctly.

4.3 Challenges of Implementing Cryptographic Standards

In addition to implementation practices, we asked participants to share common challenges and obstacles when implementing cryptographic standards.

Backward Compatibility are a Challenge in Standard Updates. About half of the participants reported difficulties with the adoption of standard updates. Participants were mainly concerned about the trade-off between the improved security of an update and the potential impact of breaking changes in their product. Some participants even did not adopt security updates at all: *“The major killer thing about an update for us is if you might have too much trouble with older clients.”* (P5). Participants reported deprecating outdated components of a cryptographic standard in an implementation as a major challenge. Participants voiced varying opinions regarding updates and deprecation. Overall, a few participants reported that they need to consider standards updates only rarely. However, this is still a concern for some crypto developers. One participant remarked on the rift between having to work with long-term deployments of computer systems and the constant developments and updates of standards:

“Then again, this is a bit of an annoyance of mine, is that with SCADA stuff, you're looking at lifetimes of 10, 20, 30 years. Whereas with the crypto guys, nothing's ever static, nothing ever stays the same. There's a constant [...]and] pointless churn in that a lot of stuff gets updated not because there's any particular reason to, but because we've just got a new flashy thing and we need to update it, which makes it incredibly difficult.” — P20

While the discovery of vulnerabilities will not wait for timespans this long, these types of deployments require legacy and backward-compatibility support, which standard organizations need to provide. Furthermore, participants working on multiple standards reported that handling backward compatibility can be more difficult for some standards than others. One participant gave the example that if an existing OpenPGP-encrypted email uses an outdated cryptographic primitive that is not supported by the recipient's OpenPGP client, that message becomes unreadable. Therefore, a standard update might result in a breaking change that cannot easily be addressed.

Algorithm Variations and Complexity. The complexity of cryptographic standards adds further challenges for implementers. Often, standards support multiple cryptographic primitive, algorithm, or protocol variations with different levels of security as a tradeoff for performance, storage requirements, or additional, optional features. For example, SHA-3 [57] supports different variations, such as SHA3-224, SHA3-256, SHA3-384, and SHA3-512, providing tradeoffs in security and performance. Some participants reported challenges when implementing all variations, often resulting in focusing on simplicity, as one participant puts it:

“I’m trying to implement this as step two, but for now it’s important to me that I create the things that are considered necessary, or rather translate them, and then add the optional components as required.”
— P2

Other participants preferred increased security and lower attack surfaces in their implementations by deploying fewer options. Some participants cited standards’ complexity in terms of comprehensibility, length of standard documents or contradicting documents:

“If you want to implement DNSSEC today, you have to read 50 different RFCs that contradict each other. If you want to implement [Hashing to Elliptic Curves] you have one document with all the possible curves, so it’s really inconsistent.” — P8

This also feeds into the debate of openness and review that we explore in [Section 4.4](#). One participant summarized their wishes:

“[...] all I wanted was just something saying ‘This is the standard from now on, key exchange, let’s do that’ and, instead, there’s just more choices being produced, there’s less clarity around them, there are fewer eyes on all of these candidates because there are so many of them.” — P12

Our findings imply that multiple variants and options in cryptographic standards can skew the cryptographic implementation process, reduce the efficacy of reviews, and weaken the overall security of standards and implementations.

Handling Errors and Typos in Standards. Participants reported their experiences with vulnerabilities and errors in standards, where they usually find them, and how they proceed when finding one. About half of the participants gave us examples of errors in standards, from typos and small contradictions to insecure random number generators. Smaller issues, such as typos, can cause problems: *“We collected quite some remarks on just typos which can be deadly in the standard document”* (P4).

Participants shared their experiences with discovering and reporting vulnerabilities, and reported differences in feedback of standard authors or maintainers.

“For example, with S/MIME, again, it’s very easy, just email the standards authors or the people involved and you can get a fix done. [...] There doesn’t seem to be much point in submitting any change request to something like the TLS standards group, because it’ll just get ignored.” — P20

As a positive example of reporting and fixing a significant vulnerability, P9 reported:

“The first thing when talking about vulnerabilities, one of MITRE’s things is to try and support this CVE and CWE system to actually have public reporting of discovered vulnerabilities. MITRE is certainly definitely directly interested in that and has been even outside of [cryptographic field] for a while.” — P9

This demonstrates trust in organizations like MITRE, which records, reports, and publishes vulnerabilities and errors in standards, enabling proper disclosure and handling even when the standard organization is not open to feedback.

Laws and Patents Hinder Cryptography. Overall, participants had various concerns related to laws and patents regarding standards and the adoption of cryptography.

Participants addressed regulations on the export of cryptography, e.g., to foreign countries. While participants saw issues that export laws could create, this highly depended on each case. For public implementations, such as open source, export laws were reported to have changed for the better: *“[Export laws] used to be a much bigger problem. [...] for the last 20 years basically you can publish it [open source] without any problems.”* (P3). However, in other scenarios export laws can be challenging, e.g., when cryptography is part of a physical product: *“Yes, crypto is still a problem for export. We really do have to think about where can it ship our products.”* (P5). Fittingly, P17 outlined that cryptographic code itself is not physical and the concept of exporting does not make sense:

“They always talk about exporting a product, which is, I think, the idea that you have a box that was first here and that it’s there and it’s not here anymore, and this is just not how cryptographic code, a particular open source cryptographic code gets developed. The concept doesn’t fit there.” — P17

About half of our participants discussed the challenges of regulations within individual countries. Our participants also reported challenges for implementers when governments specify atypical cryptography that must be used. For example, a participant reported the need to support the Chinese ShangMi (SM) cipher suites in TLS [75] to comply with Chinese laws. As illustrated by some participants, regulations are seemingly politically motivated:

“Many countries want their own national cipher. Korea has SEED. China has SM2, 3, 4. Russia has GOST. It’s almost like a point of pride. I can see, for example, why Russia and China historically would not want to use what the US agencies picked. [...] It’d be great if we were all just one big, international happy family.” — P16

As in the case of China, such restrictive country laws can cause implementer concerns and insecurities: *“What happens if you ship something to China?”* (P5). For all the mentioned issues, a few participants stated that a legal team is needed to handle laws; some reported having access to such legal support.

Participants also mentioned a general aversion against standard-related patents. Some participants explicitly stated that they try to avoid any patented cryptography; as one fittingly summarized: *“I try to stay away from the patented [cryptography]. I try to leave a buffer zone so that even if courts are stretching the patents, I stay away from that.”* (P3). Overall, about most of the participants agreed that patents are a major problem in cryptography. The most common reason given by some participants is that patented cryptography would not be widely used: *“If you patent something in cryptography, nobody will use it for about 20 years.”* (P1). A few participants explicitly expressed their wishes for open and free cryptography standards that are not restricted by patents. Fittingly, the participants perceived that public and open cryptographic standards have taken over and replaced proprietary ones in the past. One participant who was involved in cryptographic standardization mentioned that they purposely chose not to patent their approach: *“It was absolutely clear we wouldn’t get any patent for this because what would happen? People wouldn’t use it.”* (P5). Additionally, a few participants reported that the legal overhead caused by patents hinders the implementation of cryptographic standards. A few participants further mentioned other issues regarding participants, such as concerns about patent law infringement liabilities, or consequences for individual open-source developers in patent lawsuits. Participants discussed a few cases where code was not publicly released by the person or company to avoid patent issues. Contrary to the challenges above, a few participants mentioned that they do not consider laws and patents at all, and a few participants stated that patents are a non-issue in academic open-source implementations:

“The good thing is that when you’re implementing something in an academic context, you don’t need to worry too much about patents. I can write code, I can put the code online, and that’s okay.” — P17

Besides the aforementioned practical issues, two participants outlined ethical concerns of patenting cryptography, as most cryptography would be developed by publicly-funded researchers:

“There is one thing I think I don’t like when publicly funded researchers apply for patents. I think this is ethically more than questionable. [...] I think cryptography is a science of communication that requires interoperability and that requires that everyone can use these things. Patents are killers for cryptographic standards.” — P4

Despite all critiques regarding patents, one participant mentioned that patents are the main incentive for industry contributions, but directly relativized the point by mentioning their companies’ bad experiences with patents in the context of cryptographic standards.

The reservations towards patents might also be caused by accessibility issues. A few participants stated various challenges, such as finding, keeping up-to-date with, and knowing all relevant patents—or when a patent expires. Similarly, a few participants mentioned associated costs as a barrier to implementing patented cryptography. Therefore, a few participants mentioned that patents are mainly a thing that lawyers should care about: *“Actually, pretty much all people in the industry don’t talk about patents. The first rule, I think that they all get told like: leave that to the lawyers.”* (P17).

4.4 Suggestions for More Implementation-Friendly Standards

After discussing challenges, we asked participants for suggestions to improve future cryptographic standardization.

Preference for Public or Open-Source Approaches. Our participants mentioned the criticality and impact of openness in cryptographic standards and the standardization process. In our interviews, we identified four areas: openness in standardization, open community around standards, mistrust through lack of openness, and a rift between open, often academic, standards, and closed industry use.

First, regarding openness in standardization, almost all mentioned it as an important aspect of cryptography. Of these, most mentioned that standardization bodies should be transparent and also make standards publicly accessible. For our participants, openness begins with the standardization process. They appreciate open standardization processes and in particular named the NIST cryptography standard competitions and the general standardization process of IETF as positive and trustworthy examples, as they enable public review of standards and therefore foster trust:

“If there’s a closed standard, I almost never look at it. I think that’s not going to be something that people have done any real security review. Maybe there are ten people who looked in a room at some ISO standard, but it’s not the same as a publicly written, publicly developed standard. NIST has its standards online. That’s something that makes

them much more likely for people to look at them and review them. I'll look for those.” — P3

Second, when implementing standards, some actively rely on the community around open standards, e.g., public forums of standard committees, mailing lists, and the ability to contact standard authors and contributors with questions, to implement their cryptography securely. One particular advantage for large implementations is the ability to implement along released drafts and address implementation problems during the standardization process before the standard is finalized, as illustrated by one participant regarding ECH, an extension of TLS.

“The implementation [of ECH] started when the spec was still relatively fluid. It was changing. There’s a GitHub repo with the spec where there’s a bunch of issues are processed there. There are a bunch of issues that were processed during the various meetings and mailing lists. The implementation [...] started early and [there] was feedback between the implementations and the specification at that point.” — P6

The ability to control and discuss changes before a version of the standard is finalized aligns with the problems participants reported regarding standard updates and changes (see [Section 4.3](#)).

Third, missing openness and transparency might induce mistrust. Almost all participants mentioned actively avoiding paid or inaccessible standards. As one participant put it:

“I hate paid standards. I hate having to pay to read research papers because sometimes you know that this is something that could be useful to your protocol or whatever you are working on, and you just can’t access it. [...] If I have to pay for that, I’m going to ignore something that I might be at risk of and that’s not cool.” — P8

Furthermore, some participants gave specific examples of standardization bodies they mistrust because of lacking transparency in their standardization processes. Particular examples mentioned the International Organization for Standardization (ISO) closed committee of decision makers and the national fast track [25], that they provide for quicker standardization requests, as sources of mistrust. They also criticized more minor instances of lacking transparency, like NIST’s final decision process in the selection of PQC algorithms being made behind closed doors, or IETF’s tendency to make decisions in IETF meetings with missing documentation and reasoning, which can exclude people from the standardization process if they are unable to attend these meetings.

Finally, a few participants also mentioned the impact of closed environments, a rift between the open academic community working on cryptographic papers and standards, and the more closed-off industry consumers.

“We end up in a situation where government and academia are the creators of standards, but industry, the consumer, and there’s a misalignment of incentives. I would like there to be a way to include the actual consumers of the standards in the feedback process more easily.” — P19

Participants reported that NDAs and closed implementations can make obtaining feedback and voicing opinions in public standardization forums difficult.

Communication and Social Components. Our participants discussed the potential to improve the current state of communication within the cryptographic standard community. Some mentioned the importance of existing explanations to some extent. First, several participants requested that reasons and explanations for decisions in cryptographic standards should be provided in the standard document for both cryptographic implementers and researchers:

“[...] requires a rationale for all the points. In some cases, it helps implementers. In other cases, for example, the opaque standards where the authors could not explain why the standards did some of these things. Some people came back and asked: ‘Why is this requirement in there?’ And the standard’s authors couldn’t explain why [...]. It helps implementers, and in the case of some standards bodies, it helps guide them in not creating standards that have incomprehensible features.” — P20

There were also mentions that those reasons should exist, but in a different place and not in the actual description of the cryptographic scheme:

“A rationale that virtually no IETF standards ever have. Because if you’ve got a standard and there’s some ambiguous section in there and there’s a rationale, you can say, okay, they meant this. Without a rationale, you have absolutely no idea. Which is why I mentioned the IEEE standards earlier: because they include a rationale, so you can see what the thinking was behind this thing.” — P20

Interviewees had opinions about the current existing “toxic” [their wording] discourse around standards. They reported that strong social involvement is required to push cryptography. Simultaneously, this social component ensures that newcomers, less involved implementers, and small groups may be discouraged from participation:

“I would like to see a body that’s a little bit more formal than IACR or the CFRG that represents some kind of broader consensus of academic cryptographers without a competitive nature coming to the fore. People who are good cryptographers somehow always have to have big egos as well.” — P6

4.5 General Trends in Cryptographic Standards

Beyond the topics mentioned above, participants also mentioned several trends in cryptography. We describe them as general tendencies among our participants for reference.

Importance of Post Quantum Cryptography. PQC is currently the strongest trend in crypto. Most participants mentioned working on PQC or watching the current PQC process. This trend is likely inflated by recruiting from the NIST PQC Forum mailing list, but since that only contributed a few participants, the trend is still clearly visible.

Required Financial Aids. On the organizational side, a few participants, as part of their final “wish”-question, explicitly asked for more financing to cover issues and recommendations we have talked about throughout the interview. Examples included open standards, resolving patent and licensing issues, and formal verification. This demonstrates the need for proper financing to develop well-working standards with high adoption rates and usability, and informs the priorities of our recommendations.

5 Discussion

Below, we discuss our findings and make recommendations for future cryptographic standards.

First, we report the wishes and needs of our interviewees mentioned (see Section 4). We extract them from the results, summarize them, and illustrate their criticality.

Second, we assess how current major cryptographic standards implement those wishes and needs. Therefore, we systematically analyzed the 21 openly available cryptographic standards our participants mentioned (cf. Table 3). Additionally, they mentioned two proprietary standards, SM4 (ISO/IEC 18033-3:2010/Amd 1:2021) and IEEE 802.11. We excluded both due to significant procurement costs and our participants’ rejection of proprietary standards. In a systematic analysis of each standard document and the linked references, we assessed how well the standards address the wishes and needs of our participants, as we illustrate below. Two authors analyzed all standard documents and the linked references independently, then met to resolve disagreements. Table 3 summarizes and illustrates the analysis findings.

Finally, we identify gaps between our participants’ wishes and needs and the major standards, and we provide our recommendations for improving future cryptographic standards.

Reference Implementations. Our participants illustrated the criticality of reference implementations when implementing standards (cf. Section 4.2). As one participant explained: “Reference implementations are great when they are readable, written in simple language like Python. That’s super useful.” (P8). Reference implementations provide examples of implementation details that may be missing in the standard

Table 3: Summary of our assessment of how well current major standards implement participant wishes and needs.

Standard	Org	Last Release	Reference Implementations	Third-Party Implementations	Test Vectors Provided	Formally Verified Standards	Verified Implementations	Supports Developer Choice	Update Notes Provided	Patent and IPR Disclosure	Open Standardization Process	Contact Information Provided	Feedback Encouragement	Bug Report Channels Provided
X.509 (RFC)	RFC	2024	½	0	●	○	○	○	○	○	○	○	○	○
Signal	De Facto	2024	1	0	○	○	○	○	○	○	○	○	○	○
Kyber (ML-KEM)	NIST (d)	2023	2+	2+	●	○	○	○	○	○	○	○	○	○
Sphincs+ (SLH-DSA)	NIST (d)	2023	2+	2+	●	○	○	○	○	○	○	○	○	○
AES	NIST	2023	1	0	●	○	○	○	○	○	○	○	○	○
(EC/Ed)DSA	NIST	2023	1	0	○	○	○	○	○	○	○	○	○	○
DNSSEC	RFC	2021	0	0	○	○	○	○	○	○	○	○	○	○
GOST	RFC (i)	2020	0	0	○	○	○	○	○	○	○	○	○	○
Wireguard	De Facto	2020	2+	0	○	○	○	○	○	○	○	○	○	○
OpenSSH	RFC	2020	1	0	○	○	○	○	○	○	○	○	○	○
S/MIME	RFC	2019	½	0	○	○	○	○	○	○	○	○	○	○
CHACHA20	RFC	2018	1	0	●	○	○	○	○	○	○	○	○	○
TLS 1.3	RFC	2018	1	0	●	○	○	○	○	○	○	○	○	○
TLS 1.2	RFC (o)	2018	½	0	○	○	○	○	○	○	○	○	○	○
RADIUS	RFC	2017	0	0	○	○	○	○	○	○	○	○	○	○
PKCS: RSA	RFC	2016	½	0	○	○	○	○	○	○	○	○	○	○
Keccak (SHA-3)	NIST	2015	2+	2+	●	○	○	○	○	○	○	○	○	○
SHA-2	NIST	2015	0	0	○	○	○	○	○	○	○	○	○	○
MD5	RFC	2011	1	0	○	○	○	○	○	○	○	○	○	○
OpenPGP	RFC	2009	0	0	○	○	○	○	○	○	○	○	○	○
SEED	RFC (i)	2005	½	0	○	○	○	○	○	○	○	○	○	○

● Fully applies. ○ Partially applies. ○ Does not apply.
 x Not applicable. ½ Parts of an implementation included or linked.
 2+ Multiple implementations for different audiences included or linked.
 (i) Informational RFC (o) Obsolete RFC (d) NIST Standard Draft

document and a starting point for implementers. Additionally, reference implementations may be formally verified. Furthermore, participants suggested documenting reference implementations better, adding more context and explanations to make implementations easier. As mentioned by our participants and in related work [16] this lack may hinder or even prevent standard adoption.

Seven of the 21 standards we analyzed provided or linked to one external reference implementation. Four included multiple reference implementations, mainly limited to modern NIST standardization efforts and WireGuard. Five provided only partial reference implementations. However, some standards lack reference implementations and only provide mathematical foundations or pseudocode. Five standards did not provide or reference any code, including older NIST standards such as SHA-2 [56] and DSA [53].

Reference Implementations Should be Mandatory. We recommend making reference implementations and their documentation part of the standard drafting phase. NIST, for example, required multiple implementations as part of its PQC and SHA-3 standardization efforts and linked to them on their accompanying website. We also recommend updating existing standards and including links to reference implementations.

Third-Party Implementations. Similar to reference implementations, our participants highlighted the criticality of third-party implementations. They confirm compatibility with the general ecosystem surrounding a standard. As one participant mentioned, implementation requirements, e.g., as realized by the IETF, help standards to gain adoption, collect public feedback, and provide an essential point of reference. Third-party implementations might be a better starting point for the industry to implement cryptographic standards than reference implementations, considering both simple reference and formally proven implementations. For example, third-party implementations might focus more on efficiency beyond functionality and correctness.

Among the 21 analyzed standards, only the three NIST standardization efforts provided third-party implementations (see [Table 3](#)). They linked to them on each of the accompanying websites that were part of the submission, including the code and contact information.

Multiple Third-Party Implementations Should be Referenced. We recommend standardization bodies to require multiple third-party implementations as part of their standardization effort. However, maintaining a list of third-party implementations inside the standard may be unsustainable. Therefore, we recommend following the example of the NIST SHA-3 and SLH-DSA standards and providing links to third-party implementations on their accompanying website.

Test Vectors. Our participants mentioned test vectors as an effective tool to ensure compliance with standards. They required test vectors to confirm the correctness of their implementations, check edge cases, and perform negative testing for mistakes. Participants reported generating them using reference or third-party implementations in case of missing test vectors.

Twelve of the standards we analyzed provided test vectors either as part of the standard or referenced in the standard document. NIST, for example, provides test vectors for many of their cryptographic primitives at one central location on their websites [51]. And TLS 1.3 includes test vectors in a separate RFC [72]. Test vectors are missing in seven other cases, including major de facto standards like the Signal protocol and WireGuard. However, the reference implementations seemed sufficient to generate high-quality test vectors.

More and Negative Test Vectors. We suggest standard documents should include a set of test vectors to help developers check functional implementation correctness. We also recommend in-

cluding negative test vectors whenever applicable to better support testing for common implementation mistakes.

Formally Verified Standards. Proofs for standards also increase trust in standards, making their algorithms more likely to be adopted and errors more straightforward to identify across implementations. However, as participants confirmed, generating proofs for standards is a complex and time-consuming process; therefore, we want to encourage formal verification but argue that it should not be mandatory.

Only three of the standards we assessed provided formal verification. These are the NIST standardization efforts SHA-3, Kyber, and Sphincs+, along with TLS 1.3, which reference multiple research publications to prove their security. The remaining standards lacked formal verification.

Make the Current Verification Trend a Requirement for Future Standards. Formal verification was provided in the most recent standards we analyzed. In line with participants' wishes, we recommend continuing this trend and including proofs in standard documents, reference papers, or similar publications. We recommend considering updating highly critical existing standards to add formal verification.

Verified Implementations. Verified implementations can extend formal verification. Like the proof of security guarantees for a standard, verified implementations prove that a specific implementation complies with a standard. As our participants reported, these implementations are used to test other implementations' compatibility or to generate custom test vectors. As the code might be difficult to read, it should not be the only type of implementation.

Only two of the standards we assessed provided formally verified implementations. TLS 1.3 refers to a formally verified implementation. The WireGuard documents link to formal verification for parts of the implementations. The remaining standards do not provide formally verified implementations.

Offering Verified Implementations Can be Beneficial. While providing verified implementations takes significant effort, their benefits for implementers of cryptographic standards are high. Some participants suggested using verified implementations as a black box binding for other implementations, removing the need for custom implementations from scratch. Hence, we highly recommend verified implementations as mandatory for future standardization processes.

Supporting Developer Choices. As participants reported, many cryptographic standard documents are challenging to read and comprehend, as standards are often complex, lengthy, and not targeted towards implementers. They may encompass multiple documents and provide multiple versions, e.g., SHA-3 has six variants [57], and SSH consists of 23 RFCs, five drafts, and, depending on compatibility goals, additional OpenSSH-specific protocol extensions [62]. Identifying the critical parts for an implementation is often not straightforward and complicates the implementers' lives even further.

Hence, our participants expressed a high demand for better organization of standard documents and improved readability and comprehensibility.

Seven of the 21 standards we analyzed, mostly modern RFC standards, were well-structured and contained helpful summaries for implementers. NIST provides a quantifiable security rating from one to five in their newest standards [58], which are helpful indicators for implementers. The in-depth breakdowns of RFC contain important hints at pitfalls in security and implementation issues. Eleven standards contained some form of summary or overview to support implementation. They also described standards' strengths and weaknesses that were otherwise difficult to grasp.

Helpful Summaries and Security Tips are Highly Beneficial for Adoption. Since cryptographic standards can quickly encompass a high level of complexity that implementers cannot easily deal with, we recommend cryptographic standard documents transparently communicate different versions, provide actionable hints and summaries, and illustrate strengths and weaknesses.

Provide Update Notes. For updates, participants reported difficulties maintaining legacy support. This aligns with the findings of related work [20] and demonstrates the need to consider update paths as a core property of modern standards.

In our dataset, twelve standards provided easy-to-find and explicit update notes. RFC errata and draft updates provide changes in the style of git diffs; Eight cases, however, including more significant changes like TLS 1.2 to 1.3 and changes in NIST standards, focused on updates in prose.

Concise Update Notes Significantly Help Implementers. We recommend future standards to provide diff-style update notes when standard documents change. Update notes should address general update information for implementers and discuss common pitfalls.

Patent and Intellectual Property Rights Disclosure. Another common obstacle was legal uncertainty. Participants did not feel able to address country-specific laws adequately in international open-source projects. Patents or intellectual property rights (IPRs) often discouraged our participants and were negatively associated with license fees. Modern standards tend to prefer cryptography that is not weighted down by IPRs, as stated by NIST [59] and IETF [9] guidelines for standard proposals. One participant put it this way: *“in general, when somebody tries to patent cryptography, it’s actually more of a deterrent than if it was just, standardized, made public and allowed broadly.”* (P11).

In our analysis, 17 of 21 standards contained information on patents or IPR disclosures. We noticed that Kyber contains a patent waiver, and older standards like RSA and ECDSA had patents applied after their standardization. IETF standards provide a collection of IPR disclosures in their data tracker [28]. The IETF also requires patents to be disclosed and prefers standards not encumbered by known patents. This

allows implementers to assess the patent situation around a standard quickly.

IPR and Patents Should be Reported Alongside Standard Documents. We notice the trend towards open standards with no IPR rights attached and patents that apply. For IPRs in general, we advocate reporting them in one central location, e.g., as provided by the IETF data tracker.

Open Standardization Process. Our participants preferred open standardization processes such as NIST and IETF with transparent requirements, public decision-making, and involvement. Our participants say open processes contribute to more easily sustainable and trustworthy standards. Standardization processes, like for AES, were praised for their openness, and efforts like the formal verification of TLS 1.3 increased the trust in those standards. Our participants' suggestions align with the principles laid out by Hao [22].

Analyzing the 21 standards, we found eleven standards implemented open standardization processes. These include NIST competitions and IETF requests for comments and draft phases. Six de facto standards, some of which IETF and NIST later published in open form (e.g., SHA-2, MD5, ECDSA), had no initial open standardization processes.

Open Standards and Standardization to Create Trust. While this is already the case for many standards, we recommend that both the standards and the standardization process should be open. This fosters trust and is a critical adoption factor. IETF and NIST have made great efforts to provide open processes, and we recommend that they be further refined.

Contact Information. Our participants illustrated the importance of contact information for standard maintainers regarding implementation details or other questions. They sometimes consult experts or standard authors to help with gaps or uncertainties in standards. Our participants also perceived contact information as beneficial for general feedback and bug reporting.

In our standard analysis, 20 provided contact information; only the more legacy SHA-2 lacked them in the document. IETF's Datatracker [28] works well in identifying responsible parties and offering contact information and the ability to disclose errata for all RFCs, even though in the case of de facto standards like OpenSSH, it may be unclear which organization between the OpenBSD project, OpenSSH authors, or the IETF, is responsible.

Provide Clear Contacts for Standards. Future standards should adhere to established practices in existing standards regarding providing contact information. For example, IETF's Datatracker quickly allows finding contact information and prevents confusion about whom to contact for which purpose.

Feedback Encouragement. Our participants perceived communication with standard bodies and the community as challenging. For example, organizations ignored bug reports,

demotivating further engagement. However, participants also mentioned the presence of solid egos and the requirement of extensive subject knowledge that seems to reign within standardization processes. Champions who led the discussion emerged due to their experience and dedication to the field of cryptography. While these champions are undoubtedly essential to drive public standardization processes, especially in recognizing and arguing with stakeholders like state-level attackers attempting to weaken cryptographic standards, the strict discourse around these standards can scare newer or less dedicated contributors away.

14 of the 21 analyzed standards actively encouraged feedback from the broader community, including all IETF standards. This finding highlights the community-driven benefits of open standardization processes. However, only the PQC draft standards explicitly encouraged providing active feedback for newcomers or stakeholders outside the community.

Standards Community Should be More Open to Newcomers.

As a few participants explicitly stated, standardization efforts should implement encouraging processes to involve newer or smaller voices.

Bug Report Channels. Finally, participants reported issues with reported typos or other minor mistakes that standard maintainers did not take seriously, leading to participants not reporting them anymore.

In our standard analysis, only three standards—OpenSSH, Signal, and WireGuard—provided clear instructions for bug reporting and encouraged it. These standards reference their sources and their bug trackers as clear points for bug reporting.

Transparently Communicate Bug Reporting Instructions and Processes is Important.

We recommend transparent and open communication channels for bug reporting. They should include feedback error reports to support developers and encourage error reporting, particularly for newcomers or outsiders to cryptographic standardization processes.

6 Conclusion

In this work, we presented and discussed the findings from interviewing 20 cryptographic software developers regarding the implementation process of cryptography based on standards. We inquired about their experiences and handling of various phases of the implementation process. Overall, we find that standards can be complex and lengthy, and they sometimes lack summaries and support that implementers prefer. This includes official reference implementations, the availability of test vectors, and documentation. Furthermore, while cryptographic software is largely already openly available, we found that proprietary software or patents are often frowned upon, even by industry participants, as they raise legal issues, obscure standards, and hinder standard adoption.

Acknowledgments

We thank the cryptography community and our participants for their invaluable contributions to our work. We also thank the anonymous reviewers, especially our shepherd, for their constructive feedback and support in improving the paper. Finally, we thank our students, Daniel Kabuth and Thassilo Schiepaniski, for their help with early text drafts, project scouting, and interview conception as part of our lecture. This research was funded by the Volkswagen-Stiftung Niedersächsisches Vorab – ZN3695 and the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972. This work was also supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/211/OHK3/3T/18 funded by the MEYS of the Czech Republic.

Availability

We provide a replication package containing supplementary resources, including invite emails and the consent form, the interview guide, the anonymized final codebook, and a list of sampled open-source software projects. It is available under the following link: <https://doi.org/10.17605/OSF.IO/3H95W>.

References

- [1] Yasemin Acar, Michael Backes, Sascha Fahl, Simson Garfinkel, Doowon Kim, Michelle L. Mazurek, and Christian Stransky. “Comparing the Usability of Cryptographic APIs”. In: *Proc. 38th IEEE Symposium on Security and Privacy (SP’17)*. IEEE, 2017.
- [2] José Bacelar Almeida, Manuel Barbosa, Gilles Barthe, Benjamin Grégoire, Adrien Koutsos, Vincent Laporte, Tiago Oliveira, and Pierre-Yves Strub. “The Last Mile: High-Assurance and High-Speed Cryptographic Implementations”. In: *Proc. 41th IEEE Symposium on Security and Privacy (SP’20)*. IEEE, 2020.
- [3] Harald T. Alvestrand. *A Mission Statement for the IETF*. 3935. RFC Editor, 2004.
- [4] Amberscript. *Amberscript: Audio & Video Transcription*. URL: <https://www.amberscript.com/en/> (visited on 02/08/2024).
- [5] Erinn Atwater, Cecylia Bocovich, Urs Hengartner, Ed Lank, and Ian Goldberg. “Leading Johnny to Water: Designing for Usability and Trust”. In: *Proc. 11th Symposium on Usable Privacy and Security (SOUPS’15)*. USENIX, 2015.

- [6] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. “*The SPHINCS+ Signature Framework*”. In: *Proc. 26th ACM Conference on Computer and Communication Security (CCS’19)*. ACM, 2019.
- [7] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. 5280. RFC Editor, 2008.
- [8] Scott O. Bradner. *The Internet Standards Process – Revision 3*. 2026. RFC Editor, 1996.
- [9] Scott O. Bradner and Jorge Contreras. *Intellectual Property Rights in IETF Technology*. 8179. RFC Editor, 2017.
- [10] Victoria Clarke and Virginia Braun. “*Thematic Analysis*”. In: *Encyclopedia of Critical Psychology*. New York, NY: Springer New York, 2014, pp. 1947–1952.
- [11] Dave Crocker, Eric Burger, and Russ Housley. *Reducing the Standards Track to Two Maturity Levels*. 6410. RFC Editor, 2011.
- [12] Quoc Huy Do, Pedram Hosseyni, Ralf Küsters, Guido Schmitz, Nils Wenzler, and Tim Würtele. “*A Formal Security Analysis of the W3C Web Payment APIs: Attacks and Verification*”. In: *Proc. 43rd IEEE Symposium on Security and Privacy (SP’22)*. IEEE, 2022.
- [13] Jason A. Donenfeld. *Formal Verification - Wireguard*. URL: <https://www.wireguard.com/formal-verification/> (visited on 06/10/2024).
- [14] RFC Editor and Heather Flanagan. *RFC Style Guide*. 7322. RFC Editor, 2014.
- [15] Xinguo Feng, Yanjun Zhang, Mark Huasong Meng, and Sin G. Teo. “*Detecting Contradictions from CoAP RFC Based on Knowledge Graph*”. In: *Network and System Security*. Ed. by Xingliang Yuan, Guangdong Bai, Cristina Alcaraz, and Suryadipta Majumdar. Cham: Springer Nature Switzerland, 2022.
- [16] Konstantin Fischer, Ivana Trummová, Phillip Gajland, Yasemin Acar, Sascha Fahl, and Angela Sasse. “*The Challenges of Bringing Cryptography from Research Papers to Products: Results from an Interview Study with Experts*”. In: *Proc. 33rd Usenix Security Symposium (SEC’24)*. USENIX, 2024.
- [17] Google. *HTTPS encryption on the web – Google Transparency Report*. URL: <https://transparencyreport.google.com/https/overview?hl=en> (visited on 10/23/2023).
- [18] Google. *wycheproof*. URL: <https://github.com/google/wycheproof> (visited on 02/08/2024).
- [19] Peter Leo Gorski, Yasemin Acar, Luigi Lo Iacono, and Sascha Fahl. “*Listen to Developers! A Participatory Design Study on Security Warnings for Cryptographic APIs*”. In: *Proc. CHI Conference on Human Factors in Computing Systems (CHI’20)*. ACM, 2020.
- [20] Harry Halpin. “*SoK: Why Johnny Can’t Fix PGP Standardization*”. In: *Proc. 15th International Conference on Availability, Reliability and Security (ARES’20)*. ACM, 2020.
- [21] Julie M Haney, Mary Theofanos, Yasemin Acar, and Sandra Spickard Prettyman. “*“We make it a big deal in the company”: Security Mindsets in Organizations that Develop Cryptographic Products*”. In: *Proc. 14th Symposium on Usable Privacy and Security (SOUPS’18)*. USENIX, 2018.
- [22] Feng Hao. “*Prudent Practices in Security Standardization*”. In: *IEEE Communications Standards Magazine* 5.3 (2021), pp. 40–47.
- [23] Wes Hardaker. *Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP)*. 6353. RFC Editor, 2011.
- [24] Sven Hebrok, Simon Nachtigall, Marcel Maehren, Nurullah Erinola, Robert Merget, Juraj Somorovsky, and Jörg Schwenk. “*We Really Need to Talk About Session Tickets: A Large-Scale Analysis of Cryptographic Dangers with TLS Session Tickets*”. In: *Proc. 32nd Usenix Security Symposium (SEC’23)*. USENIX, 2023.
- [25] John L. Hill. “*Publicly available specification. A new paradigm for developing international standards*”. In: *Computer* 28.10 (1995), pp. 97–98.
- [26] C. Huitema, J. Postel, and S. Crocker. *Not All RFCs are Standards*. RFC 1796. RFC Editor, 1995.
- [27] IETF. *IETF 118 Meeting Agenda*. URL: <https://datatracker.ietf.org/meeting/118/agenda> (visited on 11/08/2023).
- [28] IETF. *IETF Datatracker*. URL: <https://datatracker.ietf.org/> (visited on 05/24/2023).
- [29] IETF. *saag Info Page*. URL: <https://www.ietf.org/mailman/listinfo/saag> (visited on 02/09/2024).
- [30] Internet Engineering Task Force. *IETF | Internet Engineering Task Force*. URL: <https://www.ietf.org/> (visited on 10/23/2023).
- [31] Jan Jancar, Marcel Fourné, Daniel De Almeida Braga, Mohamed Sabt, Peter Schwabe, Gilles Barthe, Pierre-Alain Fouque, and Yasemin Acar. “*“They’re not that hard to mitigate”: What Cryptographic Library Developers Think About Timing Attacks*”. In: *Proc. 43rd IEEE Symposium on Security and Privacy (SP’22)*. IEEE, 2022.

- [32] Matthias J. Kannwischer, Peter Schwabe, Douglas Stebila, and Thom Wiggers. “Improving Software Quality in Cryptography Standardization Projects”. In: *Proc. 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&P)*. IEEE, 2022.
- [33] Arun Kanuparthi, Hareesh Khattri, Parbati Kumar Manna, Narasimha Kumar V Mangipudi, and Parbati K. Manna. *Use of a Cryptographic Primitive with a Risky Implementation*. 2020. URL: <https://cwe.mitre.org/data/definitions/1240.html> (visited on 05/02/2023).
- [34] Erin Kenneally and David Dittrich. *The menlo report: Ethical principles guiding information and communication technology research*. Aug. 2012.
- [35] Katharina Krombholz, Karoline Busse, Katharina Pfeiffer, Matthew Smith, and Emanuel von Zezschwitz. “If HTTPS Were Secure, I Wouldn’t Need 2FA” - End User and Administrator Mental Models of HTTPS”. In: *Proc. 40th IEEE Symposium on Security and Privacy (SP’19)*. IEEE, 2019.
- [36] Adam Langley, Wan-Teh Chang, Nikos Mavrogiannopoulos, Joachim Strombergson, and Simon Josefsson. *ChaCha20-Poly1305 Cipher Suites for Transport Layer Security (TLS)*. 7905. RFC Editor, 2016.
- [37] Adam Langley, Mike Hamburg, and Sean Turner. *Elliptic Curves for Security*. 7748. RFC Editor, 2016.
- [38] David Lazar, Haogang Chen, Xi Wang, and Nickolai Zeldovich. “Why does cryptographic software fail?: a case study and open problems”. In: *Proc. 5th Asia-Pacific Workshop on Systems (APSys’14)*. ACM, 2014.
- [39] J. Lazar, J.H. Feng, and H. Hochheiser. *Research Methods in Human-Computer Interaction*. Elsevier Science, 2017.
- [40] Yong Li, Yuanyuan Zhang, Juanru Li, and Dawu Gu. “iCryptoTracer: Dynamic Analysis on Misuse of Cryptography Functions in iOS Applications”. In: *Network and System Security*. Ed. by Man Ho Au, Barbara Carminati, and C.-C. Jay Kuo. Cham: Springer International Publishing, 2014.
- [41] WhatsApp LLC. *Message Privately | Whatsapp Features*. URL: <https://www.whatsapp.com/privacy?lang=en> (visited on 10/23/2023).
- [42] Chris M. Lonvick and Sami Lehtinen. *The Secure Shell (SSH) Protocol Assigned Numbers*. 4250. RFC Editor, 2006.
- [43] Chris M. Lonvick and Tatu Ylonen. *The Secure Shell (SSH) Transport Layer Protocol*. 4253. RFC Editor, 2006.
- [44] Nora McDonald, Sarita Schoenebeck, and Andrea Forte. “Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice”. In: *ACM on Human-Computer Interaction 3.CSCW*, 72 (2019), pp. 1–23.
- [45] Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk. “Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E)”. In: *Proc. 30th Usenix Security Symposium (SEC’21)*. USENIX, 2021.
- [46] Kamyar Mohajerani, Luke Beckwith, Abubakr Abdulgadir, Eduardo Ferrufino, Jens-Peter Kaps, and Kris Gaj. *SCA Evaluation and Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process*. Cryptology ePrint Archive, Paper 2023/484. <https://eprint.iacr.org/2023/484>. 2023.
- [47] Kathleen Moriarty, Burt Kaliski, and Andreas Rusch. *PKCS #5: Password-Based Cryptography Specification Version 2.1*. 8018. RFC Editor, 2017.
- [48] Nicky Mouha and Christopher Celi. *A Vulnerability in Implementations of SHA-3, SHAKE, EdDSA, and Other NIST-Approved Algorithms*. Ed. by Mike Rosulek. Springer International Publishing, 2023.
- [49] National Institute of Standards and Technology. *National Institute of Standards and Technology*. URL: <https://www.nist.gov/> (visited on 10/23/2023).
- [50] Yoav Nir and Adam Langley. *ChaCha20 and Poly1305 for IETF Protocols*. 8439. RFC Editor, 2018.
- [51] NIST. *Cryptographic Standards and Guidelines - Examples with Intermediate Values*. URL: <https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/example-values> (visited on 05/21/2023).
- [52] NIST. *Cryptographic Standards and Guidelines Development Process*. URL: <https://csrc.nist.gov/Projects/crypto-standards-development-process> (visited on 05/30/2023).
- [53] NIST. *FIPS 186 Digital Signature Standard (DSS)*. U.S. Department of Commerce, 2023.
- [54] NIST. *FIPS 197 Advanced Encryption Standard (AES)*. U.S. Department of Commerce, 2001.
- [55] NIST. *Post-Quantum Cryptography – Selected Algorithms 2022*. 2022. URL: <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>.
- [56] NIST. *Secure Hash Standard (SHS)*. U.S. Department of Commerce, 2015.
- [57] NIST. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. U.S. Department of Commerce, 2015.

- [58] NIST. *Stateless Hash-Based Digital Signature Standard*. U.S. Department of Commerce, 2023.
- [59] NIST. *Submission Requirements and Evaluation Criteria for the Post-Quantum Cryptography Standardization Process*. URL: <https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/call-for-proposals-final-dec-2016.pdf> (visited on 02/08/2024).
- [60] NIST *pqc-forum - Google Groups*. URL: <https://groups.google.com/a/list.nist.gov/g/pqc-forum> (visited on 02/09/2024).
- [61] Marten Oltrogge, Nicolas Huaman, Sabrina Klivan, Yasemin Acar, Michael Backes, and Sascha Fahl. “Why Eve and Mallory Still Love Android: Revisiting TLS (In)Security in Android Applications”. In: *Proc. 30th Usenix Security Symposium (SEC’21)*. USENIX, 2021.
- [62] OpenSSH. *OpenSSH Specifications*. URL: <https://www.openssh.com/specs.html> (visited on 06/10/2024).
- [63] OpenBSD Project. *OpenSSH Project History*. URL: <https://www.openssh.com/history.html> (visited on 05/21/2023).
- [64] Sazzadur Rahaman and Danfeng Yao. “Program Analysis of Cryptographic Implementations for Security”. In: *Proc. 2017 IEEE Secure Development Conference (SecDev’17)*. IEEE, 2017.
- [65] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. 8446. RFC Editor, 2018.
- [66] Joran Dirk Greef of Ronomon. *ChaCha20-Poly1305 with long nonces*. URL: <https://www.cve.org/CVERecord?id=CVE-2019-1543> (visited on 05/02/2023).
- [67] Dominik Schürmann. *History - OpenPGP*. URL: <https://www.openpgp.org/about/history/> (visited on 05/21/2023).
- [68] Signal. *Is it private? Can I trust it? – Signal Support*. URL: <https://support.signal.org/hc/en-us/articles/360007320391-Is-it-private-Can-I-trust-it-> (visited on 10/23/2023).
- [69] Signal. *Technical information*. URL: <https://signal.org/docs/> (visited on 05/21/2023).
- [70] sobolevn. *Awesome Cryptography*. URL: <https://github.com/sobolevn/awesome-cryptography> (visited on 02/08/2024).
- [71] Christian Stransky, Dominik Wermke, Johanna Schrader, Nicolas Huaman, Yasemin Acar, Anna Lena Fehlhaber, Miranda Wei, Blase Ur, and Sascha Fahl. “On the Limited Impact of Visualizing Encryption: Perceptions of E2E Messaging Security”. In: *Proc. 17th Symposium on Usable Privacy and Security (SOUPS’21)*. USENIX, 2021.
- [72] Martin Thomson. *Example Handshake Traces for TLS 1.3*. 8448. RFC Editor, 2019.
- [73] Luke Valenta, Nick Sullivan, Antonio Sanso, and Nadia Heninger. “In Search of CurveSwap: Measuring Elliptic Curve Implementations in the Wild”. In: *Proc. 3rd European Workshop on Usable Security (EuroUSEC’18)*. The Internet Society, 2018.
- [74] Joanne Woodage and Dan Shumow. “An analysis of NIST SP 800-90A”. In: *Proc. Advances in Cryptology (EUROCRYPT’19)*. Springer, 2019.
- [75] Paul Yang. *ShangMi (SM) Cipher Suites for TLS 1.3*. 8998. RFC Editor, 2021.

A Cryptographic Standardization

Cryptographic standards describe cryptographic primitives and protocols tested and approved through a central entity, a standardization body. This approach has become increasingly important for cryptography [54, 57, 55]. Below, we give an overview of classes of cryptographic standards critical for our work and provide an overview of standardization bodies.

A.1 Cryptographic Standards

Open cryptographic standards should contain relevant implementation details, potentially different security options, the effect of these options, and, ideally, requirements for the systems implementing these cryptographic standards. For our work, we distinguish two cryptographic standards types:

Cryptographic Primitives. These describe the low-level building blocks of cryptography. We generally refer to these as the building blocks or primitives that can be used in security protocols to secure information. Examples include hash functions (e.g., the SHA-2 family [56]), key exchanges like X25519 [37], and key derivation functions like PBKDF2 [47]. We also include components or processes that describe combinations of the primitives above to provide new capabilities but still apply to the general purpose of securing information. Examples include the AES block cipher and the post-quantum-cryptography algorithm SPHINCS+ [6] presenting an approach to enable One-Time Signatures to be combined into a general purpose signature scheme that is resistant against quantum attackers.

Cryptographic Protocols. Cryptographic protocols use and combine the primitives as mentioned above to secure specific use cases and larger systems, and to prevent unintentional leaks. Standardized protocols also ensure that all participants using a protocol can securely communicate with each other, independent of the concrete implementation a participant uses. Examples include TLS, used to secure TCP connections or the Web Authentication (WebAuthn) and FIDO APIs/protocols enabling passwordless authentication via browser. Our definition of cryptographic protocols also includes many *de facto* standards introduced by popular and widely available software without an explicit standardization body beyond the software developers. Prominent examples of these de facto standards include OpenSSH as a secure remote shell protocol, WireGuard [13] as a secure virtual networking layer, and the Signal protocol [69] for end-to-end encrypted messaging.

A.2 Standardization Bodies

One particularly well-known standard body for cryptography is NIST. They introduce Federal Information Processing Standards (FIPS), which are a collection of public standards that NIST developed for computer systems of (non-military) U.S. government agencies and their contractors. For example, FIPS standardizes the Advanced Encryption Standard (AES) in FIPS 197 [54] and the Digital Signature Standard (DSS) including Rivest-Shamir-Adleman cryptosystem (RSA), Elliptic Curve Digital Signature Algorithm (ECDSA), and recently Edwards-curve Digital Signature Algorithm (EdDSA) in FIPS 186 [53].

Next is the IETF, proposing RFC as their version of standards. The goal of the IETF, according to RFC 3935 [3], is to provide protocol standards, best practices and other information intended to influence how people design, use and manage the internet. RFCs are formally published by the so-called *RFC Editor* after a public review process. Anyone with sufficient experience can contribute to and support internet standards created by the IETF within their defined processes [14], making this standardization body and its processes open and accessible. Examples for internet standards provided by the IETF are TLS in RFC 6353 [23] and proposed standards like the ssh authentication protocol [43].

For the modern Internet, the most important standard body is the World Wide Web Consortium (W3C), responsible for the standards surrounding HTML.⁶ Finally, there are standard bodies like the ISO,⁷ the American National Standards Institute (ANSI),⁸ and the International Telecommunication Union (ITU) offering industry standards in the larger context of IT and other aspects of cooperation worldwide.

⁶<https://html.spec.whatwg.org/> (Accessed 2024-05-27)

⁷<https://www.iso.org/home.html> (Accessed 2024-05-27)

⁸<https://ansi.org/> (Accessed 2024-05-27)

A.3 Standardization Process

There are various processes to standardize cryptographic primitives and protocols. While creating a standard document has been unified for each organization, deciding what to standardize has changed a lot recently.

Public Standardization Effort. This first approach was recently adopted by NIST for their PQC effort, SHA-3, and AES [52]. They post public requirements for a new standard. Then everyone can propose cryptographic primitives and protocols, out of which a few winners are chosen by a panel of experts. Then public feedback for that selection is collected in multiple rounds until a final standard can be drafted.

Public Comments on Standards. Here, standards are more publicly decided on mailing lists and in public meetings. The organization and contributors work on drafts of an open standard as part of so-called *Request for Comments (RFCs)* until it reaches maturity, often starting with a suggested algorithm. This is the common approach of IETF and W3C.

De Facto Standards. For this third approach, a protocol is proposed and used often enough by industry or academia to become a standard, a de facto standard. It addresses a gap in existing standardization and becomes so successful that one of the organization bodies later translates it into a conventional standard. Examples of this are the Ed25519 primitive that has been added to FIPS in 2023 [53], after being known and used [62, 65] for multiple years. Other examples are the OpenSSH and OpenPGP standards, standardized by the IETF as part of open implementations of de facto standard proprietary software [63, 67].

Standard Through Organization. Finally, there is the conventional process used by ISO, ANSI, and, in earlier years, by NIST, where a private panel of experts selected by the organization works on and later publishes a standard. Backdoors and attacks on such standards have caused a need for publicly evaluated and verified cryptography and communication standards. Because of this, even originally closed standards with a large adoption in modern cryptography, like X.509 for certificates, have obtained open variants [7]. This closed approach is also used by some government agencies for government mandated cryptography, e.g., GOST and Seed that are then published as informational RFCs, outside the IETF's typical standard efforts [26], and usually produce de facto standards with geographically restricted use.