Why Johnny Signs with Next-Generation Tools: A Usability Case Study of Sigstore

Anonymous author(s)

***Abstract*—Software signing is the most robust method for ensuring the integrity and authenticity of components in a software supply chain. However, traditional signing tools place key management and signer identification burdens on practitioners, leading to both security vulnerabilities and usability challenges. Next-generation signing tools such as Sigstore have automated some of these concerns, but little is known about their usability and adoption dynamics. This knowledge gap hampers the integration of signing into the software engineering process.**

To fill this gap, we conducted a usability study of Sigstore, a pioneering and widely adopted exemplar in this space. Through 18 interviews, we explored (1) the factors practitioners consider when selecting a signing tool, (2) the problems and advantages associated with the tooling choices of practitioners, and (3) practitioners' signing-tool usage has evolved over time. Our findings illuminate the usability factors of next-generation signing tools and yield recommendations for toolmakers, including: (1) enhance integration flexibility through officially supported plugins and APIs, and (2) balance transparency with privacy by offering configurable logging options for enterprise use.

I. Introduction

The reuse of software components in modern products creates complex supply chains and expands the attack surface for these systems [1]–[3]. A core strategy for mitigating this risk is to establish provenance—verifying actor authenticity and artifact integrity—through software signing [4], [5]. Software signing is the strongest guarantee of provenance [6] and is widely advocated by academia, industry consortia, and government regulations [7]–[9]. High-profile supply-chain attacks—such as the SolarWinds breach—have further underscored the need for stronger, more automated signing infrastructure. In response, tools have evolved into two classes: *traditional solutions* (*e.g.,* OpenPGP [10]), which delegate key management and signer identification to practitioners, and *next-generation, identity-based tools* (*e.g.,* Sigstore [11]), which further automate the process via short-lived keys and external identity providers.

Tooling has significantly advanced various technological fields [12]–[15] and has often played a key role in the adoption of these technologies [16], [17]. There is a continual need to evaluate the adoption practices and usability of such tools to inform practitioners, potential users, researchers, and toolmakers of their effectiveness in different contexts [18]. Traditional software signing tools have been studied in this way, most notably in several "Why Johnny Can't Encrypt" inspired studies [19]–[21]. Although next-generation signing tools promise greater automation, there remains a lack of comprehensive usability and adoption analyses for these solutions. Such studies could assess how automation affects adoption suitability, elicit practitioners' desired improvements, and guide future tool design and research.

Our work offers the first examination of next-generation software signing tools' usability as a critical factor influencing their adoption and practice. Building on and improving previous methodologies — which relied on novice users and summative usability assessments criticized for their lack of actionable feedback [18], [22] — we conducted interviews with 18 experienced security practitioners to study current software signing tools, focusing on Sigstore as a case study. We analyzed our data with a formative usability framework [23], focusing on two usability dimensions: practitioners' experiences with Sigstore and its usability relative to other signing tools. From these interviews, we distilled three key themes: the evolution and motivations behind tool adoption, the limitations and challenges of currently adopted tools, and the specific implementation practices shaping real-world signing workflows.

Our results highlight the usability factors practitioners consider when adopting next-generation software signing tools. This serves as formative feedback for tool designers and researchers, offering insights into the drivers of tool usage. We present these findings through a clear articulation of practitioners' challenges and the merits of using particular tools. Additionally, we found that practitioners' decisions to adopt a software signing tool are influenced by less commonly discussed factors, such as the tool's community, relevant regulations and standards, and certain functionalities of next-generation Software signing tools.

In summary, we make the following contributions:

1) We report on the difficulties and advantages of using Sigstore, a next-generation software signing tool.
2) We discuss how practitioners' choices of software signing tools change over time and the factors influencing those decisions.

Significance of Study: Our work targets Sigstore—a pioneer next-generation signing tool that has seen rapid uptake across major open-source ecosystems [24]–[26], cloud-native projects, and corporations [27]–[31], involving a large number of engineers worldwide. By unpacking the usability factors that drive or hinder Sigstore's adoption, we not only inform the ongoing refinement of its automation workflows (e.g., certificate issuance and key management) but also provide a template for evaluating and improving similar Next-generation(identity-based) signing solutions. Also, since Sigstore exemplifies a broader class of tools that embed security into automated build

and deployment processes, our findings can be potentially generalized to guide the design and implementation of other automated software-supply-chain technologies, ultimately advancing the usability—and thus the security—of automated software engineering practices.

II. Background & Related Works

In this section, we review software signing technologies (§II-A) and usability assessments (§II-B).

A. Software Signing

Software signing ensures the authorship and integrity of a software artifact by linking a maintainer's cryptographic signature to their software artifact. The implementation of software signing is shaped by several factors: the software artifacts to be signed (*e.g.,* driver signing [32]), the root-of-trust design (*e.g.,* public key infrastructure/PKI [33]), and ecosystem policies (*e.g.,* PyPi [34] and Maven [35]).

1) Software Signing Tools: Software signing tools can be classified into two groups: traditional and next-generation (identity-based) [5]. Traditional signing tools such as GPG [36] follow a conventional public-key cryptography approach, relying on long-lived asymmetric key pairs and delegating key-management responsibilities to the signer. Trust is established through endorsements by other signers.

Next-generation (identity-based) signing tools automate key management by issuing short-lived, ephemeral certificates tied to a signer's identity via external OpenID Connect (OIDC) or OAuth providers. This "keyless signing" approach — exemplified by Sigstore [11], OpenPubKey [37], and SignServer [38] — eliminates long-term key handling and embeds transparency through append-only logs, improving usability and security. We discuss the architecture and workflow for next-generation signing tools in §IV-B using the case of Sigstore.

2) Effects of Signing Tools on Software Signing Adoption: Previous studies recognize that better tooling influences software-signing adoption. For example, Schorlemmer *et al.* [6] performed a measurement study of several software package registries, identifying tooling as a key determinant of signature quality. Similarly, Kalu *et al.* [39] explore implementation and adoption frameworks in commercial settings, highlighting tool usability, compatibility, and integration challenges. While these works emphasize broad usability factors, they offer little detailed guidance on what makes a signing tool usable, overlooking the fine-grained technical and workflow requirements that drive real-world tool adoption.

B. Usability Studies in Software Signing

1) Usability – Definition, Theory & Evaluation Frameworks: Tools enable engineers to adopt new software development practices [40]. A usable tool is one whose functionality effectively supports its intended purpose—achieving product goals [18], [41]. Studies assessing implementation strategies of tools are typically framed as usability evaluations [42].

To evaluate a tool's usability, studies typically follow a usability evaluation framework [42]. There are two kinds: summative, which measures a tool's effectiveness [43], [44], and formative, which provides iterative feedback for improvement [18], [45]. Given the aims of this work, we took a formative approach. We considered several formative frameworks [46]–[49] and found that Cresswell *et al.'s* four-factor framework [23] helped explain our data the best. That framework is summarized in Figure 1.

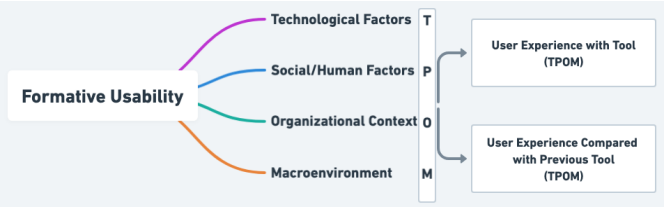

Fig. 1. Cresswell's Usability Framework [23]. It assesses usability based on four factors. *Technological (T)* concerns the technical characteristics of the system (*e.g.,* performance, ease of use) and data management (availability, integrity). *Social/Human (P)* relates to users and their interactions with the tool, *e.g.,* user satisfaction, engagement, prior experience, and correct use of features. *Organizational Context (O)* are factors internal to participants' organization, *e.g.,* training and leadership support. *Macroenvironmental (M)* are external influences, *e.g.,* regulations, economics, and vendor relations.

2) Empirical Studies: The usability of traditional signing tools—often studied in the context of encryption—remains a significant challenge. Whitten and Tygar's seminal study, "*Why Johnny Can't Encrypt"* [19], found PGP 5.0's interface too complex for non-experts, a problem that persisted in PGP 9 [20], especially around key verification, transparency, and signing. Subsequent work [50]–[52] has underscored the public-key model's inherent complexity. Research comparing manual versus automated signing tools has yielded mixed results: some works observed no clear usability advantage [53], [54], whereas Atwater *et al.* showed a user preference for automated PGP-based solutions [55]. Although informative, these studies focus on traditional encryption scenarios (*e.g.,* email and messaging), limiting their applicability to broader software-signing workflows.

These studies exhibit weaknesses in their scope, methodology, and the relevance of their findings to software signing. First, they primarily focus on evaluating the usability of signing *tooling* in the context of signing communication (*e.g.,* email, messages), limiting their application to signing software (*i.e.,* securing the software supply chain). Furthermore, much of this research is outdated, reflecting decades-old technologies and failing to incorporate contemporary practitioner experiences. Their methodology predominantly employs summative usability assessments that rely on novice volunteers, which limits the applicability of their findings for guiding tool enhancements [56], [57].

III. Knowledge Gaps & Research Questions

Existing literature on software signing (see §II) has examined practice and adoption across private and open-source ecosystems, highlighting the importance of tool choice in signature quality. Prior usability studies have focused exclusively on traditional signing tools and have been limited to evaluating

user-interface design. However, next-generation signing tools remain understudied, creating a need for:

- A holistic evaluation of how software signing tools influences software-signing adoption and practice, and how usability affects this process.
- Practitioner perspectives on the nuanced role of tooling in shaping software-signing use and practice. Current research (see §II-B2) is limited because it primarily relies on experiments with novice users, while studies in §II-A2 use quasi-experiments or general practitioner interviews without a specific focus on signing tools.

In light of these gaps, a comprehensive usability evaluation of next-generation signing tools would offer valuable insights into their role in adoption, improve their design, and enhance their adaptability.

Research Questions: Building on identified gaps in next-generation tooling usability and its impact on adoption, we frame our research around Sigstore—our chosen next-generation signing tool case study—and ask:

RQ1: What strengths and weaknesses of Sigstore influence its adoption in practice?

RQ2: What factors influence practitioners' decisions to change the software signing tools they use?

IV. Methodology

We selected a case study method to answer our RQs. Our methodology is illustrated in Figure 2. This section provides our design rationale (§IV-A), defines the case study context (§IV-B), discusses instrument design (§IV-C), data collection (§IV-D), and analysis (§IV-E).

Our Institutional Review Board (IRB) approved this work.

A. Rationale for Case Study

The case study is a research method used to investigate a phenomenon within its real-life context [58]–[61]. We selected the case study method for our research because it allows for an in-depth exploratory examination of our chosen case, enabling a deeper understanding of the usability factors that influence the adoption of a software signing tool in practice, rather than a broad but superficial cross-analysis [62]. In selecting Sigstore, we adhered to the criteria outlined by Yin [59] and Crowe *et al.* [60]. Specifically, we considered the intrinsic value and uniqueness of the case study unit (high), access to data (adequate), and risks of participation (low). In §VI we discuss how our findings may generalize.

B. Case Study Context – Sigstore

Following guidelines outlined by Runeson & Höst [58], we describe the importance of our selected case study context and the technical details of the system.

Use in Practice: Sigstore launched in 2021. It provides keyless, ephemeral keys and signer authentication mechanisms to simplify the signing process used by traditional tools. Within 13 months of its public release, Sigstore recorded 46 million artifact signatures [25]. It has significant adoption and

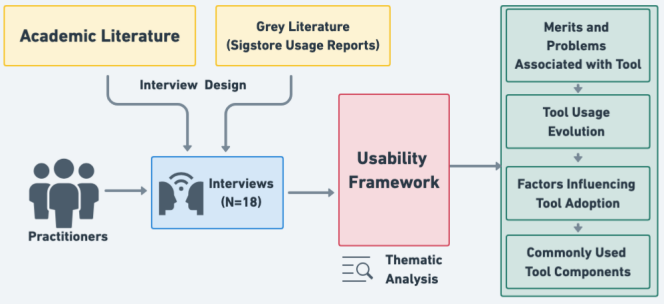

Fig. 2. Study Methodology. We developed our interview protocol by reviewing Sigstore grey literature and the academic literature. Finally, we analyzed our results using a thematic analysis informed by Cresswell's formative usability evaluation framework (Figure 1).

support from open-source ecosystems (*e.g.,* PYPI [26], Maven [24], NPM [63]) and many companies (*e.g.,* Autodesk [31], Verizon [29], Yahoo [64]). Thus, Sigstore is a major next-generation software signing platform, making it a suitable context for examining usability in modern signing tools.

Components & Workflow: We detail Sigstore's architecture and signing workflow, to familiarize the reader with this example of next-generation software signing tools, and to interpret participants' statements referring to these components.

Sigstore's primary components are [11]:

- **Cosign:** CLI and Go library to create, verify, and bundle signatures and attestations.
- **OIDC Identity Provider:** OpenID Connect services (*e.g.,* GitHub, Google, Microsoft) that authenticate signers and issue short-lived identity tokens.
- **Fulcio (Certificate Authority):** Issues ephemeral signing certificates after verifying the signer's OIDC token.
- **Rekor (Transparency Log):** Append-only ledger that records each signing event and certificate issuance.
- **Monitors and Verifiers:** Services or tools that validate the signer's identity, certificate status, and signature integrity.

Figure 3 shows the Sigstore software signing workflow. First, the signer authenticates with an OIDC provider to obtain a short-lived token. Next, Fulcio verifies this token and issues a one-time signing certificate. The signer then uses Cosign to apply that certificate to the software artifact. Finally, Rekor logs the certificate and signature, and verifiers retrieve entries from both Fulcio's log and Rekor to confirm the signer's identity and ensure the certificate and signature remain valid.

Commonality with other Next-Gen Tools: Sigstore, like other next-generation signing tools such as OpenPubKey [37], AWS Signer [65], and SignServer [38], shares core functionality in signer identity management and automated key handling. However, unlike Sigstore, most of these tools do not include a built-in certificate authority (*e.g.,* OpenPubKey) or a transparency log (*e.g.,* OpenPubKey, SignServer). Instead, such services must be added via external integrations. While our focus on Sigstore allows for an in-depth case analysis, many of its design principles—such as identity-based authentication and ephemeral keys—are shared by its peers, suggesting

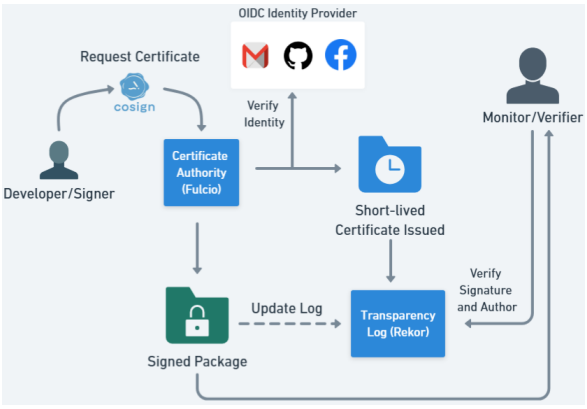

Fig. 3. Sigstore Signing Workflow. The software author requests a certificate from a certificate authority, which confirms the signer's identity through an identity provider. The signature and signer's identity are recorded in a transparency log, which the verifier can monitor to confirm the validity of the signature upon downloading the signed package. See Appendix K for legible figure

potential generalizability across next-generation signing tools

C. Instrument Design & Development

Following guidance from Saldaña [66], we used semi-structured interviews to examine various aspects of tooling usability, *e.g.,* implementation challenges, perceived benefits, and comparison to other signing tools. This approach let us pose consistent questions across all subjects, with the ability to probe about unique aspects of each subject's circumstances.

To develop our interview instrument, we did not bind our questions to any prior usability framework. Sigstore is a nascent technology and in such contexts an exploratory design is recommended [58], [59]. Only after this exploratory phase did we map our findings onto a formal usability framework, ensuring it reflected practitioners' lived experiences rather than constraining data collection to preconceived categories. Therefore, to construct the interview protocol, we:

1) Reviewed grey literature—Sigstore blogs [67] and usage reports [28]–[31], [68]–[71]——to ground our questions in real-world implementations and updates.
2) Used a snowball review of top security–conference works on signing primitives [11], [72] and empirical studies (see §II), to identify gaps and avoid redundant questions.

Next, we refined our interview protocols through a series of practice and pilot interviews [73]. We first conducted two practice interviews with the secondary authors of this work, followed by pilot interviews with the first two interview subjects. For example, we moved the question about each participant's team's major product to the demographics section to provide context for their responses. Our full interview instrument consisted of three main topics (B-D). Topics B and C did not fit into the same analysis framework as Topic D. Thus, following the guidance of Voils *et al.* [74], we analyzed those topics in another work [BLINDED].[1] In this work, we focus on the data for Topic D.

[1]That work is blinded for review. We attest no duplication of material.

TABLE I
SUMMARY OF INTERVIEW PROTOCOL. DATA FROM TOPICS B AND C ARE ANALYZED IN [BLINDED]. WE ANALYZE TOPIC D IN THIS PAPER. WE PRESENT OUR FULL PROTOCOL IN **??**

Topic(# Questions)	**Sample Questions**
A. Demographics (4)	What is your role in your team?
B. Perceived software supply chain risks (4)	Can you describe any specific software supply chain attacks (*e.g.,* , incidents with 3rd-party dependencies, code contributors, OSS) your team has encountered?
C. Mitigating risks with signing (8)	How does the team use software signing to protect its source code (what parts of the process is signing required)?
D. Signing tool adoption (5)	
Question Categories in Topic D	
D1.	What factors did the team consider before adopting this tool/method (Sigstore) over others?
D2.	What was the team's previous signing practice/tool before the introduction of Sigstore?
D3.	How does your team implement Sigstore (which components of Sigstore does the team mostly use)?
D4.	Have you encountered any challenge(s) using this tool of choice?
D5.	Have you/your team considered switching Sigstore for another tool?

A summary of the questions in our interview protocol is shown in Table I. Topic D consisted of five main question categories. Questions D3–D4 probe current tooling experiences. Questions D1–D2 explore previous tools, and D5 asks about transition factors. See Appendix B for the full protocol.

D. Data Collection

Population: Our study investigates signing tools using a case study of the Sigstore ecosystem. As such, our target participant pool is centered on Sigstore users. Furthermore, our research questions, as reflected in our interview instrument, aim to understand the reasons prior to adoption that teams choose software tools (Sigstore), how Sigstore is used in signing, and the challenges and benefits of Sigstore.

To effectively answer these questions related to decision-making, we required a target population of high-ranking or experienced practitioners who either oversee Software signing compliance or manage the infrastructure of their respective teams or organizations.

To recruit this target population, we employed a non-probability purposive and snowball sampling approach [75]. We leveraged the personal network of one of the authors to send an initial invitation to members of the 2023 KubeCon (hosted by the Linux Foundation) organizing committee, who were also part of the In-toto [76], [77] Steering Committee (ITSC). This initial invitation yielded 6 participants. We then expanded our recruitment through snowball sampling, relying on recommendations from initial participants (5) and authors' contacts (7) to recruit an additional 12 participants. We interviewed 18 subjects, but only 17 participants could share enough details for analysis.

TABLE II
DEMOGRAPHICS. FOR ROLE, "*Senior management*" ARE SENIOR MANAGERS/DIRECTORS/EXECUTIVES; "*Technical leader*" ARE SENIOR/LEAD/PARTNER/PRINCIPAL ENGINEERS; AND "*Engineer*" AND "*Manager*" ARE JUNIOR STAFF.

ID	Role	Experience	Software Type
P1	Research leader	5 years	Internal POC software
P2	Senior mgmt.	15 years	SAAS security tool
P3	Senior mgmt.	13 years	SAAS security tool
P4	Technical leader	20 years	Open-source tooling
P5	Engineer	2 years	Internal security tooling
P6	Technical leader	27 years	Internal security tooling
P7	Manager	6 years	Security tooling
P8	Technical leader	8 years	Internal security tooling
P9	Engineer	2.5 years	SAAS security
P10	Engineer	13 years	SAAS security
P11	Technical leader	16 years	Firmware
P12	Technical leader	4 years	Consultancy
S13	Senior mgmt.	16 years	Internal security tool
P14	Research leader	13 years	POC security software
P15	Senior mgmt.	15 years	Internal security tooling
P16	Senior mgmt.	15 years	SAAS
P17	Manager	11 years	Security tooling

Participant Demographics: Our subjects were experienced security practitioners responsible for initiating or implementing their organization's security controls or compliance. This gives them the relevant context to assess their organization's strategies for adopting software signing tools. Subjects came from 13 distinct organizations, all companies. Relevant demographics are in Table II. Thirteen participants reported their organizations use Sigstore. Two use internally developed tools and two rely on Notary v1 and PGP signing. Those four participants enrich our findings, as their experiences shed light on usability factors that *hinder* adoption.

Interviews: We conducted our interviews over Zoom. Each interview lasted ∼50 minutes, with the parts pertinent to topic D comprising ∼15 minutes (one third) of each interview. The lead author conducted these interviews. We offered each subject a $100 gift card as an incentive in recognition of their expertise.

E. Data Analysis

We transcribed our interview recordings using www.rev.com's human transcription service and anonymized all potentially identifiable information.

After transcription and anonymization, our analysis began with a data familiarization stage [78] and proceeded through four phases, carried out by two analysts.[2] In the first two phases, we inductively memoed and coded our transcripts, developing our initial codebooks without referencing our usability framework [79]. In the final phase, we derived themes from our inductively generated codes and then applied our usability framework to further refine the extracted themes. We describe these steps in detail next. We remind the reader that the transcript memoing and codebook was developed across the full interview transcripts, including Topics B and C. This manuscript focuses on the data and codes pertinent to Topic D.

[2]One of the analysts is the lead author.

1) Memoing & Codebook Creation: First, beginning with the initial 18 transcripts, the two analysts independently memoed a subset of six anonymized transcripts across three rounds of review and discussion, following the recommendations and techniques outlined by O'Connor & Joffe [80] and Campbell *et al.* [81]. This approach ensured the reliability of our analysis process while optimizing data resources.[3] After each round, emerging codes were discussed in meetings, resulting in a total of 506 coded memos. We then reviewed the coded memos and finalized an initial codebook with 36 code categories.

To assess the reliability of the coding process, we combined the techniques outlined by Maxam & Davis [82] and Campbell *et al.* [81]. Specifically, we randomly selected an unanalyzed transcript, which we coded independently. Following Feng *et al.'s* [83] recommendation, we used the percentage agreement measure to evaluate consistency, as both analysts had contributed to developing the codebook. We obtained an 89% agreement score, suggesting a stable coding scheme [80], [81]. We resolved disagreements through discussion.

2) Codebook Revision: Following the reliability assessment of our codebook, and given the substantive agreement recorded, one of the analysts proceeded to code the remaining transcripts in two batches of six transcripts per round. In the first batch, six new code categories were introduced, which were subsequently reviewed by the other analyst. The same analytical process was applied to the final six transcripts, leading to the addition of four new code categories, which were subsequently reviewed by the other analyst. Notably, of the 10 codes added during this phase of codebook refinement, only three provided new insights into the data. By mutual agreement between both analysts, these were categorized as minor codes.

To assess the adequacy of our sample size and analysis, we calculated code saturation for each category in our codebook. Following the recommendations of Guest *et al.* [84], we measured the cumulative number of new codes introduced in each interview. We saw saturation after interview 12, with no new codes after that. The results are presented in Figure 7.

3) Thematic Analysis & Usability Framework Analysis: Each interview had a median of 15 codes for topic D. Next, following the method proposed by Braun & Clarke [85], we derived initial themes from our code categories. We focused on identifying themes related to Topic D of our protocol (Table I).

We then conducted another round of analysis using a formative usability framework. We selected Cresswell's framework (Figure 1) based on the nature of our data.

V. RESULTS

First, we present a Sankey diagram illustrating the changes in signing tool usage among our participants' teams (Figure 4). Many participants (6) transitioned from using no tools to adopting Sigstore. Others transitioned from existing tools to Sigstore: 3 from GPG, 2 from Notary, 1 from Skopeo, and 2

[3]Although there is no consensus on the ideal proportion of data to use, O'Connor and Joffe suggest that selecting 10–25% is typical [80]. We randomly chose 6 transcripts (33% of data) to develop our initial codebook.

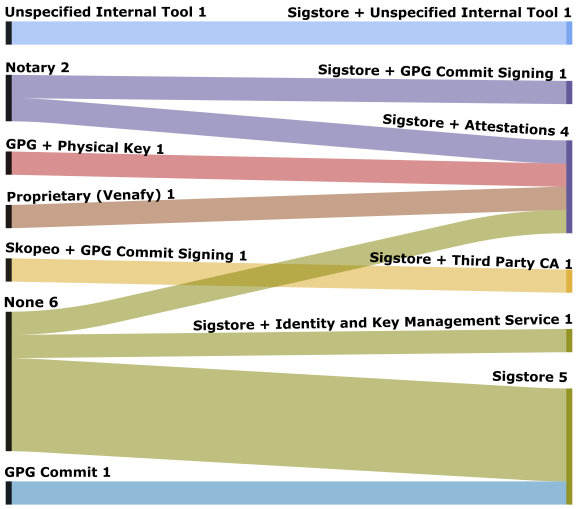

Fig. 4. Practitioners' Changes in Software Signing Tools.

from proprietary or internal tools. The final 4 subjects did *not* transition to Sigstore.

To understand why they did (and did not) change tools, we structure the results by RQ using Cresswell's framework.

A. RQ1: Strengths & Weaknesses of Sigstore

The first aim of our research work was to ascertain from the the experiences of our subjects the strengths and weaknesses they have experienced while using Sigstore.

1) Sigstore's Strength: We present a summarized list of practitioner-reported advantages of using Sigstore in Table III. First, we extracted the benefits our participants experienced while using Sigstore from their responses. These reflections capture their firsthand experiences with Sigstore.

We grouped their responses into eight categories: ease of use, identity management for signers, compatibility with emerging technologies, use of short-lived keys and certificates, implementation of a transparency log, encouragement of bundling signatures with attestations, free access to the public Sigstore instance, and reliability of the Sigstore service. When mapped to the Cresswell framework, these factors fell under two categories: Technological and Macroenvironmental.

a) Technological Factors: Ease of use: This was the most frequently discussed advantage of Sigstore over other Software signing tools. Participants commonly highlighted the simplicity of following the Sigstore workflow—particularly how key management is fully automated reduce configuration overhead. P7 *(security tools, cloud security)* states, *" I don't really know of any other options that provide the same conveniences that Sigstore provides... It really is just one command to sign something and then one more command to verify it, and so much of the work is handled by Sigstore."* Other advantages associated with *Ease of Use*, as mentioned by practitioners, include the seamless integration of the Sigstore workflow into CI/CD automation using GitHub Actions and the reduced complexity of key management.

TABLE III
SUMMARY OF PRACTITIONER-REPORTED ADVANTAGES OF USING SIGSTORE. PRACTITIONERS CITED SIGSTORE'S TECHNOLOGICAL AND MACROENVIRONMENTAL FACTORS AS ADVANTAGES.

Topics & Associated Examples	Subjects
Technological Factors	
Ease of Use	**8**
Signing Workflow & Verification	P1, P2, P7, P8, P10-13
Setting up with automated CI/CD actions	P9
No key distribution problems	P1
Use of Short-lived Keys & Certificate	**3** –P2, P3, P11
Signer ID Management	**4**
Use of OIDC(Keyless) to authenticate signers	P3, P5, P10, P12
Compatibility with Several New Technologies	**4**
Integrability with SLSA build	P12
Integrability with several container registries/technologies	P9, P10
Integrability with several cloud-native applications	P11
Precence of a Transparency Log	**3**
Transparency logs increase security	P5, P10
Evaluation of signing adoption using logs	P8
Bundling Signatures With Provenance Attestations	**2** – P3, P4
Reliability of Service	**1** – P7
Macroenvironmental Factors	
Free/Open-Source	**2** – P7, P13

Identity management for Signers: Sigstore's integration of OIDC identity management was a strength per our participants. In traditional software signing, signer identity is often an ad-hoc feature and may be optional for the signing authority. In Sigstore, however, this behavior is integrated with the Fulcio certificate authority as an identity management step before creating a signature. Participants greatly valued the elimination of the key generation and distribution phase, as it reduced the time spent distributing keys to key servers and searching for keys, streamlining the signing process. This aligns with the intuition that identity management is easier than key management for developers [11]. P16 *(SAAS, SSC security tool)* highlights this point: *"I can associate my identity with an OIDC identity as opposed to generat[ing] a key and keep[ing] track of that key...I could say, 'Oh, this is signed with [Jane]'s public GitHub identity'...that's much better"*

Compatibility with Several New Technologies: A portion of our participants reported that Sigstore's integration and compatibility with a wide range of modern technologies provided a significant advantage. The mentioned technologies included security tooling such as SLSA build attestations, as well as containerization technologies and cloud computing applications. In most cases, the resources required to set up connections to these technologies were minimal or nonexistent. P12 *(consultancy, cloud OSS security)* states, *"a lot of clients are getting into signing their containers pretty trivial to do if you use something like cosign...Sigstore is the easy choice because it just works on any registry, and integrating it to Kubernetes is pretty easy "*

Use of Short-lived Keys/Certificates: Sigstore's enforcement

of short-lived keys and certificates as a security feature was recognized by our participants as a major strength of the tooling. Traditional software signing tools require users to manually set expiration times for cryptographic elements such as signatures, keys, and certificates at the time of generation. While this approach provides customization flexibility, it also introduces the risk of long-lived cryptographic materials remaining valid for extended periods, increasing the likelihood of compromise. Participants appreciated Sigstore's automated management of key and certificate lifetimes, which mitigates this risk. P2 *(SAAS, SSC security tool)* states, *"So (with) PGP, you still have to figure out the key distribution problem. Because Sigstore uses short-lived keys that solves a lot of those issues around key storage and key distribution..."*

Presence of a Transparency Log: The ability to audit signing actions was crucial for adopters. Participants valued the transparency log in the Sigstore ecosystem because it provides a tamper-resistant record of all changes to an artifact's metadata—functionality not offered by other signing systems. This auditability gave users greater confidence in the integrity of signed artifacts.

P9 ***(SAAS, SSC security)****: "From a higher level, we're going to use a transparency log to determine what was actually signed and what was written into that transparency log."*

Other technological factors noted by participants are the reliability of Sigstore's public instance.

b) Macroenvironmental Factors: Free/Open-Source: Sigstore's open-source nature was also highlighted by participants as a key advantage. They noted that this reduces both setup and maintenance barriers, particularly for users leveraging Sigstore's public deployments.

P7 ***(security tools, cloud security)****: "I think it's amazing that Sigstore is free and public"*

2) Sigstore's Weaknessses: We summarize the difficulties reported by participants in using Sigstore in Table IV. Most of these weaknesses mapped to multiple usability factors.

a) Macroenvironmental & Organizational: Enterprise Adoption Limitations: Many of our participants mentioned that Sigstore is not suited for large enterprise applications. The issues reported by participants include rate limiting—where Sigstore restricts the number of signatures that can be created per unit time—along with latency concerns, where the service's turnaround time for a large volume of signatures is problematic. Additionally, participants highlighted the lack of dedicated support and maintenance teams due to Sigstore 's open-source nature, as well as its unsuitability for organizations in regulated sectors for the same reason.

P9 ***(SAAS, SSC security)****: "We were relying on the public Sigstore instance, and I don't think it could meet our signing needs in terms of just the capacity of signatures we needed as an enterprise [rate limit]."*

P3 ***(SAAS, SSC security tool)****: "For customers that are very large in scale, the biggest of the Fortune 100 or customers operating in very highly regulated environments, I think operating your own Sigstore instance within that air gap in a private environment could be valuable. And I think it depends on the amount of software you're producing and the frequency with which you're producing it."*

The issues here were a mixture of all usability factors.

TABLE IV
REPORTED DIFFICULTIES ENCOUNTERED BY PARTICIPANTS IN USING SIGSTORE. WE INDICATE THE ASSOCIATED USABILITY FACTOR OF EACH CATEGORY OF WEAKNESS USING – T-TECHNOLOGY, P-SOCIAL/HUMAN, O-ORGANIZATIONAL, M-MACROENVIRONMENTAL FACTORS.

Topics & Associated Examples	Subjects
Enterprise Adoption Limitations	**7 subjects**
Rate Limiting Problems – ***T***	P3, P7, P10, P11
Lack of dedicated Support & Maintenance – ***T & P***	P11, P2
Not Suited for Regulated Organizations – ***M & O***	P3
Latency Concerns – ***T***	P6
Transparency Log Issues	**6 subjects**
Not Suitable for private Setup – ***T***	P2, P3, P6, P10, P13
Use in Air Gap Conditions – ***T***	P2, P3, P8
Efforts to Monitor Logs – ***P***	P2
Private Sigstore Instance Setup	**5 subjects**
Documentation – ***P & T***	P8, P9, P6
Limited Community Support – ***M***	P11
Infrastructure Requirements & Maintenance Costs – ***T***	P5, P6
Other Documentations and Usage Information Issues – *P & T*	**3** — P1, P10, P13
Integration to Other Systems	**3 subjects**
Attestation Storage – ***T***	P1
Gitlab & Jenkins – ***T***	P8
Other Unsupported technologies – ***T***	P12
Offline Capabilities – *T*	**2** – P3, P4
Fulcio Issues	**1** – P3
Timestamping Issues – ***T***	
Fulcio-OIDC Workflow – ***T***	
Software Libraries – *T*	**1** – P7

b) Technological: Transparency Log Issues: While the transparency log bundled with Sigstore was highlighted as a major advantage by participants, it was also a significant concern. Their concerns stemmed from the public nature of the log, where sensitive artifact metadata from company assets could be exposed in a publicly accessible record. Another common issue raised was the log's usability in air-gapped [86] environments. Additionally, participants noted the manual effort required to continuously monitor and manage the log. Most of these participants were in the process of experimenting with setting up the transparency log at the time of the interviews.

P6 ***(security tooling, digital technology)****: "We've got some teams piloting Rekor, for instance, for different traceability usages...I'm all for the transparency log, but we...have to be careful who it's transparent to at what [time]."*

Setting up Private Sigstore Instance: Sigstore's customizability was a feature frequently utilized by participants. However, it was also associated with certain challenges. The primary issues identified included limited documentation on setup, scarce community usage information, and the infrastructure re-

quirements for private deployments. Issues here were primarily related to participants' need for support resources, which were macroenvironmental and social in nature.

P15 ***(internal security tooling, aerospace security)****: "There's not a big enough community of practitioners, or help to...deploy Fulcio, like on prem, or in their own infrastructure."*

Other Documentation Issues: Overall, the documentation for Sigstore, along with practical usage information and support, was found to be insufficient. Participants noted that updates to the documentation lagged behind changes in the Sigstore product and that there was a lack of clarity regarding the use of different Sigstore components. P17 *(security tooling, internet service)* highlights this, *"I would say the documentation lacks a bit. Just I think if you look at the documentation, it was updated quite a long time ago, like the ReadMe file. And so if we could have these different options saying that, okay, if you are starting over, this is how you can do rekor and Fulcio."*

Integration to Other Systems: Participants also expressed a desire for a wider range of compatible technologies beyond what Sigstore currently supports. Commonly mentioned integrations included other CI/CD platforms such as Jenkins and GitLab, as well as attestation storage databases. These limitations were often attributed to Sigstore being an emerging technology, as highlighted by P16 *(SAAS, SSC security tool)*, *"The weakness is not everything supports it. There's a lot of weaknesses around...its being a newer technology"*

Fulcio Issues: Some Sigstore issues were also tied to its certificate authority's (Fulcio) time stamping capabilities, and implementation of OIDC for keyless signing. Fulcio acts as an intermediary using OIDC identities to bind to a short-lived certificate, this was criticized by P3. P3 also commented on the lack of timestamping support in early versions of Fulcio.

P3 ***(SAAS, SSC security tool)****: "We do have some concerns about the way Fulcio operates as its own certificate authority. So we've been looking at things like OpenPubkey as something that removes that intermediary and would allow you to do identity-based signing directly against the OIDC provider. And then I think the other big thing that we did, because Sigstore originally didn't support it, was timestamping. We want to ensure that the signature was created when the certificate was valid."*

Offline Capabilities: Another reported limitation of Sigstore is the absence of offline verification capabilities using *offline keys*. This issue is related to the transparency log's inability to function in air-gapped environments. However, in this case, participants specifically desired the ability to use Sigstore's signing and verification features in offline scenarios.

Software Libraries: The programming libraries of Sigstore were also mentioned as a challenge, particularly in cases where users wanted to integrate Sigstore's signing capabilities directly into an application.

3) Effect of Identified Problems on Sigstore components used by Subjects: We assessed the effect of participants' reported problems with Sigstore on their usage of each component. By component, nine subjects use Cosign, nine use the Fulcio certificate authority, and nine use the OIDC keyless signing and identity manager. Eight use the Rekor transparency log. Only four use Gitsign and only two make use of customized components and local Sigstore deployments.

Among these, our data shed the most light on local deployments and the Rekor log. For local deployment, several participants cited difficulties with documentation and setup, and as a result, only two successfully deployed private instances, likely due to these challenges (Table IV, §V-A2). The Rekor transparency log presents a mixed case, with both high adoption and significant reported issues. Of the eight participants who used it, three were still in the pilot phase. Although Rekor was recognized as a key factor driving adoption due to its strengths, it also faced a high number of reported issues, indicating that, despite its benefits, challenges persist for some participants.

Other Components Without Mention Though the participants in the survey shared experiences with user-facing components, discussion about other elements was notably absent. In particular, there was very little mention of log witnessing [87], or monitoring [88] for either Fulcio or Rekor. This is remarkable, as transparency solutions require the existence of these parties to ensure log integrity and identify log misbehavior. We believe that targeted future work should identify possible gaps in user perception, as well as identify user stories so users verify log witnesses and monitor information.

B. RQ2: How & Why Practitioners Change Tools

1) Drivers of Adoption Among Current Sigstore Users: For participants who adopted Sigstore, we asked which factors—prior to adoption—influenced their decision (or their organization's decision) to switch. Their responses fell into three main categories: macroenvironmental factors (*e.g.,* regulations and user communities), human/social factors (*e.g.,* prior experience with signing tools), and technological factors (*e.g.,* Sigstore's unique capabilities).

Most of our participants chose to switch to Sigstore due to poor experiences with previous tools and due to perceived advantages associated with Sigstore. We summarize these factors in Table X.

Contributions to Sigstore: Being part of the Sigstore community as *contributors* was a major reason for adopting Sigstore. Some participants (*e.g.,* P6) stated that their organizations have a policy of contributing to open-source communities they intend to adopt. Others noted that their organizations were relatively new compared to Sigstore—meaning that Sigstore had already existed before these organizations were established, and some of their members had contributed to Sigstore before joining their respective organizations. Additionally, some participants mentioned that key members of their organizations were directly involved in the initial development of the Sigstore project. P14 *(POC software & security tool, cloud security)* reflects this, *"I'd say many of the early staff, the first 10 to 20 staff were involved in the Sigstore community. There's*

not just our interest as a company in a Sigstore, but there's a lot of personal ties to Sigstore."

Available Sigstore Functionalities: Sigstore's functionalities also influenced its adoption. While several participants were drawn to the transparency log (as highlighted in §V-A1).

P5 ***(internal security tool, cloud security)****: "Sigstore has other features, like you can use your OIDC identities, like your GitHub identities as well, to do the signing, [and] they also maintain a transparency log."*

Integration with existing technologies also affected their decision. As P1 *(POC software, digital technology)* said, *"Factors I considered were, ..., the existence of tools to use it."*

Regulations and Standards: The impact of regulations and standards on adopting Sigstore was significant among our participants. Some participants who previously used other tools (2) switched to Sigstore due to security frameworks and standards recommending it as a best practice. Additionally, the remaining participants (2) noted that regulations played a partial role in their decision to adopt Sigstore.

P5 ***(internal security tool, cloud security)****: "...standards actually really helped to convince everyone that we should be doing this ... But once a standard is put in place, once the requirements are set, then it just sped up the process."*

User Community & Trust Of Sigstore Creators: The large user community of Sigstore also motivated its adoption among participants. Conversely, participants P1 and P10 cited GPG's smaller user base as a drawback. Additionally, some participants expressed trust in industry consortia such as CNCF (Cloud Native Computing Foundation)

P3 ***(SAAS, SSC security tool)****: "We do defer often at the higher level projects to different foundations. So we would be more likely to trust something from the CNCF...[than] independent developer projects."*

GPG & Notary Issues: Issues associated with previously existing tools played a major role in practitioners' decisions to switch to Sigstore. GPG-based signing implementations have historically been criticized for key management challenges and usability concerns. Our participants echoed these issues, citing additional factors such as low adoption rates, steep learning curves, and limited integration with modern technologies.

Notary-related issues primarily concerned customer demands, compatibility with other technologies, and infrequent updates. Key and identity management challenges were common to both GPG and Notary. Meanwhile, users of proprietary signing tools faced difficulties with complex setup processes.

2) Barriers and Motivators Spurring Consideration of Sigstore Among Non-Sigstore Users: In this section we highlight reasons mentioned by practitioners that have stopped them (or their organization) from switching to Next-gen tools like sigstore. We also report issues experienced by these non sigstore users that have prompted a consideration (or in some cases partial adoption of these next-gen tools).

Barriers to Considering Sigstore: As highlighted in Table V, most reasons for non-adoption were organizational.

TABLE V
NON SIGSTORE USERS: REASONS FOR NOT ADOPTING SIGSTORE. THIS TABLE SUMMARIZES THE REASONS NON-USERS GAVE FOR NOT ADOPTING SIGSTORE, GROUPED BY THEIR CURRENT SIGNING TOOL AND MAPPED TO OUR USABILITY FRAMEWORK THEME.

Tool	Subjects	Reason (Topic)
Internal tool	S8, S11	(1) Third-party security risk from tool **T** – S11, S8 (2) Organizational Use cases **O** – S8 (3) Centralizing Organization's code signing **O** – S8 (4) Organization's Open Source Policy **O** – S8
Notary-V1	S10	(1) Tool owned by Organization **O** – S10
PGP	S13	(1) Sigstore Privacy concerns **T** – S13 (2) Trust in Sigstore's Maintainers **M** – S13

P8 ***(internal security tool & cloud APIs, internet services)****: "When it comes to open source projects and open source stuff, our Organization is very skeptical. Policy and everything drives very much teams away from that sort of stuff...There's also the ongoing risk of having a product that's built outside continually shipping code and being integrated into this product, where it deals with sensitive material. So we do evaluate these things, but the scrutiny against code written outside of the company is extreme, and with good reason given the context of all the supply chain stuff."*

Other notable issues arose from technical concerns, including privacy risks associated with Sigstore's public instance and the inherent risk of relying on a third-party tool maintained by open-source contributors.

Factors Motivating Consideration for sigstore: When practitioners were asked about issues with their current tools and potential fixes, most pointed to existing tooling problems as the primary driver for considering alternatives—such as incorporating next-gen features into internal tools or partially adopting new solutions. Although some subjects reported drawing design inspiration from next-generation workflows, none specifically mentioned Sigstore. P11 *(firmware & testing software, social technology)* states, *". I'm sure at some point in time it was influenced by an open source project before they adopted it and made the right tweaks internally."* Lack of transparency and integration of these tools to other platforms were also common. We summarize these factors in Table XI.

VI. DISCUSSION

A. Lessons for Next-Generation Tools

Next-generation signing tools offer unparalleled flexibility by integrating with external platforms, and services. However, our participants consistently highlighted integration challenges as a major obstacle.

1) Automated Cross-Platform Integrations: To fully leverage the flexibility of next-gen tools, practitioners need integration modules for popular CI/CD systems, registries, and orchestration platforms. Missing integration and sparse documentation forces users to implement custom connectors, increasing the risk of misconfiguration and potential security

vulnerabilities that undermine the very guarantees signing is meant to provide.

2) Need for Inter-Platform Standards: Standardization of information exchange is a standard problem for new classes of software infrastructure *e.g.,* , SBOM [89]. These problems often harm integration capabilities. Next-generation signing tools would gain from common exchange standards—for example, uniform schemas for transparency logs or identity assertions. Projects like Diverfiy [90], which harmonize identity provisioning across multiple OIDC providers, are a promising start. Similar efforts are needed to streamline data flows between signing components, transparency services, and external tooling ecosystems.

3) Verification Workflow: A key security property in a software supply chain is the transparency of actors and artifacts, which requires verifying signer identities [4]. Previous studies have highlighted challenges in signature verification across both organizational and open-source contexts [5], [39]. Next-generation tools like Sigstore integrate multiple verification methods, such as the Cosign CLI and the transparency log, but most participants still reported usability issues with the transparency log, and documentation indicates a lack of automation for other verification methods. Further automating the verification workflow would enhance the security benefits of software signing with this class of tools.

B. Usability Patterns of Sigstore vs Non-Sigstore users

Our results reveal different drives in adoption motivations and shared usability concerns between Sigstore users and non-users. Sigstore adopters cited contributions to the Sigstore project (Human/Social factor) and macroenvironmental factors—alongside negative experiences with legacy tools—as their primary drivers. In contrast, non-users emphasized organizational considerations when choosing their signing solution.

Despite these differences, both groups reported similar technological usability issues, particularly around integration with other systems and infrastructure transparency. Notably, non-users frequently highlighted signer identification challenges, whereas Sigstore users did not—underscoring the clear usability advantage of next-generation tooling.

C. Privacy Concerns

Privacy concerns emerged as a big issue among users of Sigstore's Rekor transparency log module, with 5 out of 6 participants who reported drawbacks citing privacy as a significant concern. Privacy concerns were also major issues influencing users not to consider sigstore as suitable for enterprise. This issue also affects the use of OIDC for keyless signing. While this challenge is not unique to Sigstore [72] and is a recurring issue in software signing and modern cryptography, it underscores the need for continued innovation to address privacy-preserving mechanisms that balance transparency with user privacy, ensuring broader adoption and trust in cryptographic tools like Sigstore. While works like Speranza [72] has been proposed, efforts should focus on integrating these solutions to tools like Sigstore.

VII. Threats to Validity

To discuss the limitations of our work, we follow Verdecchia *et al.*'s [91] recommendation to reflect on the greatest threats to our research. In addition, we discuss how our experiences and perspectives may have influenced our research (positionality).

Construct Validity: Cresswell's framework targets health IT, where "tooling" denotes integrated clinical systems, so applying it to software signing carries construct risks. We mitigated risks by inductively validating themes (from its core factors) against practitioner interviews, ensuring our adapted constructs reflect signing-tool usability.

Internal Validity: Threats to internal validity reduce the reliability of conclusions of cause-and-effect relationships. An additional internal threat exists in the potential subjectivity of the development of our codebook. To mitigate this subjectivity, we utilized multiple raters and measured their agreement.

External Validity: The primary bias to our work is that we recruited from a population biased toward the use of Sigstore. We viewed this as necessary in order to recruit sufficient subjects, and feel that that result — a focused treatment of usability in Sigstore — is a valuable contribution to the literature. A second threat arises from the sample size of our work, which is due to the challenges of recruiting expert subjects. We attempted a survey (details are in Appendix E) but received only a few responses, echoing the challenges of surveying specialized communities [92]. A third threat is our study's temporal scope. Interviews were conducted between November 1, 2023, and February 9, 2024, capturing Sigstore users' experiences during that period. Since then, Sigstore has introduced some changes, but not the core workflows and components. This suggests our findings remain relevant.

Statement of Positionality: The author team has expertise in software supply chain security and qualitative research methods, *e.g.,* publishing top-tier papers in software engineering and cybersecurity venues. One of the authors is a Sigstore contributor. To avoid bias, this author was not involved in developing the research design nor the initial analysis of the results, but provided insight in later analysis.

VIII. Conclusion

Software signing guarantees provenance via cryptographic primitives, reducing supply-chain attacks. Next-generation tools like Sigstore automate traditional challenges, yet their usability remains underexplored. We interviewed practitioners to examine Sigstore's usability, identifying key adoption factors. Our findings show that Sigstore's identity-based workflows improve usability, but integration complexities, privacy concerns, and organizational constraints hinder adoption. We recommend toolmakers provide official integration plugins and privacy controls for transparency logs. These insights inform automation improvements, promoting secure, usable next-generation signing in automated software engineering.

References

[1] M. Willett, "Lessons of the solarwinds hack," in *Survival April–May 2021: Facing Russia*. Routledge, 2023, pp. 7–25.

[2] Synopsys, "2024 Open Source Security and Risk Analysis (OSSRA) Report," 2024, https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html#introMenu.

[3] S. Benthall, "Assessing software supply chain risk using public data," in *2017 IEEE 28th Annual Software Technology Conference (STC)*, 2017.

[4] C. Okafor et al., "Sok: Analysis of software supply chain security by establishing secure design properties," in *ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*. Association for Computing Machinery, 2022.

[5] Schorlemmer, T. R. et al., "Establishing provenance before coding: Traditional and next-generation software signing," *IEEE Security & Privacy*, no. 01, 2025.

[6] ——, "Signing in four public software package registries: Quantity, quality, and influencing factors," in *2024 IEEE Symposium on Security and Privacy (SP)*, May 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10646801/?arnumber=10646801

[7] The White House, "Executive order on improving the nation's cybersecurity," May 2021, https://www.whitehouse.gov/briefing-room/statements-releases/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.

[8] D. Cooper, L. Feldman, and G. Witte, "Protecting Software Integrity Through Code Signing," National Institute of Standards and Technology, Tech. Rep. ITL Bulletin, May 2018. [Online]. Available: https://csrc.nist.gov/Pubs/itlb/2018/05/protecting-software-integrity-through-code-signing/Final

[9] Cloud Native Computing Foundation, "Software supply chain best practices," May 2021, https://github.com/cncf/tag-security/blob/main/supply-chain-security/supply-chain-security-paper/CNCF_SSCP_v1.pdf.

[10] OpenPGP.org, "Openpgp," https://www.openpgp.org/, 2023. [Online]. Available: https://www.openpgp.org/

[11] Z. Newman et al., "Sigstore: Software Signing for Everybody," in *ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[12] M. Shahriari et al., "How do deep-learning framework versions affect the reproducibility of neural network models?" *Machine Learning and Knowledge Extraction*, vol. 4, no. 4, 2022.

[13] M. Abadi et al., "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 2016.

[14] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux Journal*, vol. 2014, no. 239, p. 2, 2014.

[15] C. Lattner and V. Adve, "Llvm: a compilation framework for lifelong program analysis & transformation," in *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, 2004, pp. 75–86.

[16] L. Hao et al., "Studying the impact of tensorflow and pytorch bindings on machine learning software quality," *ACM Transactions on Software Engineering and Methodology*, vol. 34, no. 1, 2024.

[17] G. Rosa et al., "What quality aspects influence the adoption of docker images?" *ACM Transactions on Software Engineering and Methodology*, vol. 32, no. 6, 2023.

[18] J. R. Lewis, "Usability: Lessons learned...and yet to be learned," *International Journal of Human-Computer Interaction*, vol. 30, no. 9, 2014.

[19] A. Whitten and J. D. Tygar, "Why johnny can't encrypt: A usability evaluation of pgp 5.0." in *USENIX security symposium*, vol. 348, 1999.

[20] S. Sheng et al., "Why johnny still can't encrypt: evaluating the usability of email encryption software," in *Symposium on usable privacy and security*. ACM, 2006.

[21] S. Ruoti et al, "Why johnny still, still can't encrypt: Evaluating the usability of a modern pgp client," 2016, https://cups.cs.cmu.edu/soups/2006/posters/sheng-poster_abstract.pdf.

[22] A. Chapanis, "Evaluating ease of use," in *Proc. IBM Software & Information Usability Symposium, Poughkeepsie NY*, 1981, pp. 15–18.

[23] K. Cresswell et al., "Developing and applying a formative evaluation framework for health information technology implementations: Qualitative investigation," *J Med Internet Res*, vol. 22, no. 6, 2020. [Online]. Available: https://www.jmir.org/2020/6/e15068

[24] Sonatype, "Central publisher portal now validates sigstore signatures," https://www.sonatype.com/blog/central-publisher-portal-now-validates-sigstore-signatures, 2024, accessed: 2025-05-30.

[25] L. Hinds and H. Blauzvern, "Sigstore: Simplifying code signing for open source ecosystems," Nov. 2023. [Online]. Available: https://openssf.org/blog/2023/11/21/sigstore-simplifying-code-signing-for-open-source-ecosystems/

[26] S. M. Larson, "Python and sigstore," https://sethmlarson.dev/python-and-sigstore, 2024, accessed: 2025-05-30.

[27] S. J. Vaughan-Nichols, "Kubernetes Adopts Sigstore for Supply Chain Security," May 2022. [Online]. Available: https://thenewstack.io/kubernetes-adopts-sigstore-for-supply-chain-security/

[28] F. Kammel, "Signing and securing confidential kubernetes clusters in the cloud with sigstore," August 2022. [Online]. Available: https://blog.sigstore.dev/signing-and-securing-confidential-kubernetes-clusters-in-the-cloud-with-sigstore-aceac3034e70

[29] Sigstore, "Security by default: How verizon new business incubation uses sigstore to demonstrate provenance," November 2022. [Online]. Available: https://blog.sigstore.dev/security-by-default-how-verizon-new-business-incubation-uses-sigstore-to-demonstrate-provenance-7beed5714738

[30] ——, "Securing your software supply chain without changing your devops workflow," December 2022. [Online]. Available: https://blog.sigstore.dev/securing-your-software-supply-chain-without-changing-your-devops-workflow-e23393a5fffa

[31] ——, "Using sigstore to meet fedramp compliance at autodesk," November 2022. [Online]. Available: https://blog.sigstore.dev/using-sigstore-to-meet-fedramp-compliance-at-autodesk-6f645a920abc

[32] D. Cooper et al., "Security Considerations for Code Signing," *NIST Cybersecurity White Paper*, Jan. 2018. [Online]. Available: https://csrc.nist.rip/external/nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.01262018.pdf

[33] V. Zimmer and M. Krau, "Establishing the root of trust," *UEFI. org document dated August*, 2016.

[34] T. Kuppusamy et al., "Pep 480 – Surviving a Compromise of PyPI: End-to-end signing of packages," oct 8 2014.

[35] Sonatype, "Working with PGP signatures," https://central.sonatype.org/publish/requirements/gpg/, accessed: 2025-05-30.

[36] "The gnu privacy guard," Dec. 2024. [Online]. Available: https://www.gnupg.org/

[37] E. Heilman et al., "OpenPubkey: Augmenting OpenID Connect with User held Signing Keys," 2023, publication info: Preprint. [Online]. Available: https://eprint.iacr.org/2023/296

[38] SignServer Project, "About SignServer: Open-Source Signing Software," https://www.signserver.org/about/, 2025, accessed 26 May, 2025.

[39] K. Kalu et al., "An industry interview study of software signing for supply chain security," in *34th USENIX Security Symposium (USENIX Security 25)*. Seattle, WA, USA: USENIX Association, aug 2025.

[40] T. Bruckhaus et al., "The impact of tools on software productivity," *IEEE Software*, Sep. 1996.

[41] "Usable security: Why do we need it? how do we get it?" O'Reilly, 2005.

[42] A. R. Lyon et al., "The cognitive walkthrough for implementation strategies (cwis): a pragmatic method for assessing implementation strategy usability," vol. 2, 2021.

[43] International Organization for Standardization, "ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs) — Part 11: Guidance on usability specification and measures," Geneva, Switzerland, 1997.

[44] C. Rusu et al., "User experience evaluations: Challenges for newcomers," in *Design, User Experience, and Usability: Design Discourse*, A. Marcus, Ed. Springer, Cham, 2015. [Online]. Available: https://doi.org/10.1007/978-3-319-20886-2_23

[45] M. Theofanos and W. Quesenbery, "Towards the design of effective formative test reports," *J. Usability Studies*, Nov. 2005.

[46] T. C. Chan et al., "A practical usability study framework using the sus and the affinity diagram: A case study on the online roadshow website." *Pertanika Journal of Science & Technology*, vol. 30, no. 2, 2022.

[47] V. Shah et al., "Is my mooc learner-centric? a framework for formative evaluation of mooc pedagogy," *International Review of Research in Open and Distributed Learning*, vol. 24, no. 2, pp. 138–161, 2023.

[48] T. S. Andre et al., "Testing a framework for reliable classification of usability problems," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 44, no. 37. SAGE Publications, 2000.

[49] L. Strifler et al., "Development and usability testing of an online support tool to identify models and frameworks to inform implementation," *BMC Medical Informatics and Decision Making*, vol. 24, no. 1, p. 182, 2024.

[50] C. Braz and J.-M. Robert, "Security and usability: the case of the user authentication methods," in *Proceedings of the 18th Conference on l'Interaction Homme-Machine*, ser. IHM '06. New York, NY,

USA: Association for Computing Machinery, Apr. 2006, pp. 199–203. [Online]. Available: https://dl.acm.org/doi/10.1145/1132736.1132768

[51] S. Ruoti et al., "Confused Johnny: when automatic encryption leads to confusion and mistakes," in *Proceedings of the Ninth Symposium on Usable Privacy and Security*, ser. SOUPS '13. New York, NY, USA: Association for Computing Machinery, Jul. 2013, pp. 1–12.

[52] A. Reuter et al., "Secure Email - A Usability Study," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, M. Bernhard, A. Bracciali, L. J. Camp, S. Matsuo, A. Maurushat, P. B. Rønne, and M. Sala, Eds. Cham: Springer International Publishing, 2020, pp. 36–46.

[53] S. Fahl, M. Harbach, T. Muders, M. Smith, and U. Sander, "Helping Johnny 2.0 to encrypt his Facebook conversations," in *Proceedings of the Eighth Symposium on Usable Privacy and Security*, ser. SOUPS '12. New York, NY, USA: Association for Computing Machinery, Jul. 2012, pp. 1–17. [Online]. Available: https://dl.acm.org/doi/10.1145/2335356.2335371

[54] S. Ruoti, J. Andersen, T. Hendershot, D. Zappala, and K. Seamons, "Private Webmail 2.0: Simple and Easy-to-Use Secure Email," Oct. 2015, arXiv:1510.08435 [cs]. [Online]. Available: http://arxiv.org/abs/1510.08435

[55] E. Atwater, C. Bocovich, U. Hengartner, E. Lank, and I. Goldberg, "Leading Johnny to water: designing for usability and trust," in *Proceedings of the Eleventh USENIX Conference on Usable Privacy and Security*, ser. SOUPS '15. USA: USENIX Association, Jul. 2015, pp. 69–88.

[56] J. Kjeldskov, M. B. Skov, and J. Stage, "Does time heal? a longitudinal study of usability," in *Proceedings of the Australian Computer-Human Interaction Conference 2005 (OzCHI'05)*. Association for Computing Machinery (ACM), 2005.

[57] K. MacDorman et al., "An improved usability measure based on novice and expert performance," *International Journal of Human–Computer Interaction*, vol. 27, no. 3, 2011.

[58] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, p. 131–164, Apr. 2009. [Online]. Available: https://doi.org/10.1007/s10664-008-9102-8

[59] R. K. Yin, *Case study research and applications: design and methods*, sixth edition ed. Los Angeles: SAGE, 2018.

[60] S. Crowe, K. Cresswell, A. Robertson, G. Huby, A. Avery, and A. Sheikh, "The case study approach," *BMC Medical Research Methodology*, vol. 11, p. 100, Jun. 2011. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3141799/

[61] J. Gerring, "What is a case study and what is it good for?" *American Political Science Review*, vol. 98, no. 2, p. 341–354, May 2004. [Online]. Available: https://www.cambridge.org/core/journals/american-political-science-review/article/what-is-a-case-study-and-what-is-it-good-for/C5B2D9930B94600EC0DAC93EB2361863

[62] P. Ralph, "ACM SIGSOFT Empirical Standards Released," *ACM SIGSOFT Software Engineering Notes*, 2021.

[63] Sigstore Blog, "Announcing npm support: Public beta," https://blog.sigstore.dev/npm-public-beta/, 2024, accessed: 2025-05-30.

[64] Yahoo Inc., "Scaling up supply chain security: Implementing sigstore for seamless container image signing," 2023, accessed: 2025-05-30.

[65] Amazon Web Services, "Aws signer developer guide," https://docs.aws.amazon.com/signer/latest/developerguide/Welcome.html, 2024, accessed: 2025-05-30.

[66] J. Saldana, *Fundamentals of qualitative research*. Oxford university press, 2011.

[67] Sigstore, "Sigstore: A new standard for signing, verifying, and protecting software," https://www.sigstore.dev/.

[68] H. Blauzvern, "How sigstore quickly patched an upstream vulnerability," October 2022. [Online]. Available: https://blog.sigstore.dev/how-sigstore-quickly-patched-an-upstream-vulnerability-76ba84ef1122

[69] S. J. Vaughan-Nichols, "Kubernetes adopts sigstore for supply chain security," May 2022. [Online]. Available: https://thenewstack.io/kubernetes-adopts-sigstore-for-supply-chain-security/

[70] J. Hutchings, "Safeguard your containers with new container signing capability in github actions," December 2021. [Online]. Available: https://github.blog/2021-12-06-safeguard-container-signing-capability-actions/

[71] The Sigstore Technical Steering Committee, "Sigstore support in npm launches for public beta," https://blog.sigstore.dev/npm-public-beta/, April 2023. [Online]. Available: https://blog.sigstore.dev/npm-public-beta/

[72] K. Merrill et al., "Speranza: Usable, privacy-friendly software signing," 2023, arXiv:2305.06463.

[73] R. J. Chenail, "Interviewing the investigator: Strategies for addressing instrumentation and researcher bias concerns in qualitative research." *Qualitative report*, vol. 16, no. 1, pp. 255–262, 2011.

[74] C. Voils et al., "Methodological considerations for including and excluding findings from a meta-analysis of predictors of antiretroviral adherence in hiv-positive women," *Journal of Advanced Nursing*, 2007.

[75] S. Baltes and P. Ralph, "Sampling in software engineering research: a critical review and guidelines," *Empirical Software Engineering*, 2022. [Online]. Available: https://link.springer.com/10.1007/s10664-021-10072-8

[76] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, "in-toto: Providing farm-to-table guarantees for bits and bytes," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1393–1410. [Online]. Available: https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias

[77] in toto and T. L. Foundation, "in-toto: A framework to secure the integrity of software supply chains," https://in-toto.io/, 2023. [Online]. Available: https://in-toto.io/

[78] G. Terry, N. Hayfield, V. Clarke, V. Braun *et al.*, "Thematic analysis," *The SAGE handbook of qualitative research in psychology*, vol. 2, no. 17-37, p. 25, 2017.

[79] J. Fereday and E. Muir-Cochrane, "Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development," *International Journal of Qualitative Methods*, 2006.

[80] C. O'Connor and H. Joffe, "Intercoder Reliability in Qualitative Research: Debates and Practical Guidelines," *International Journal of Qualitative Methods*, 2020. [Online]. Available: https://doi.org/10.1177/1609406919899220

[81] J. L. Campbell et al., "Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement," *Sociological methods & research*, vol. 42, no. 3, pp. 294–320, 2013.

[82] W. P. Maxam III and J. C. Davis, "An interview study on third-party cyber threat hunting processes in the us department of homeland security," *USENIX Security*, 2024.

[83] G. C. Feng, "Intercoder reliability indices: disuse, misuse, and abuse," *Quality & Quantity*, 2014. [Online]. Available: https://doi.org/10.1007/s11135-013-9956-8

[84] G. Guest et al., "How many interviews are enough? an experiment with data saturation and variability," *Field methods*, 2006.

[85] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, 2006. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1191/1478088706qp063oa

[86] National Institute of Standards and Technology, "Air gap," https://csrc.nist.gov/glossary/term/air_gap, 2025, accessed: 2025-05-30.

[87] A. Hicks, "SoK: Log Based Transparency Enhancing Technologies," 2023. [Online]. Available: https://arxiv.org/abs/2305.01378

[88] A. Ferraiuolo et al., "Policy transparency: Authorization logic meets general transparency to prove software supply chain integrity," in *ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, 2022.

[89] B. Xia, T. Bi, Z. Xing, Q. Lu, and L. Zhu, "An empirical study on software bill of materials: Where we stand and the road ahead," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, Melbourne, Australia, May 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10172696/

[90] C. L. Okafor, J. C. Davis, and S. Torres-Arias, "Diverify: Diversifying identity verification in next-generation software signing," *arXiv preprint arXiv:2406.15596*, 2024.

[91] R. Verdecchia, E. Engström, P. Lago, P. Runeson, and Q. Song, "Threats to validity in software engineering research: A critical reflection," *Information and Software Technology*, vol. 164, 2023.

[92] C. G. Steeh, "Trends in nonresponse rates, 1952–1979," *Public Opinion Quarterly*, vol. 45, no. 1, Spring 1981.

[93] K. Kalu, T. R. Schorlemmer, S. Chen, K. A. Robinson, E. Kocinare, and J. C. Davis, "Reflecting on the Use of the Policy-Process-Product Theory in Empirical Software Engineering," ser. ESEC/FSE 2023, New York, NY, USA, Nov. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3611643.3613075

[94] J. Cappos, S. Thomas, J. J., T. DeCleene, A. Atkins, and D. David, "The update framework (tuf)," https://theupdateframework.io, 2021, accessed: 2025-05-30.
[95] H. Sharp, H. Robinson, and M. Woodman, "Software engineering: community and culture," *IEEE Software*, 2000.

APPENDIX

OUTLINE OF APPENDICES

The appendix contains the following material:

- Appendix A: Traditional Software Signing Workflow.
- Appendix B: The interview protocol.
- Appendix C: The codebooks used in our analysis, with illustrative quotes mapped to each code.
- Appendix D: Code Saturation Analysis.
- Appendix E: Survey results, omitted from the main paper due to low quality data.
- Appendix G: Expanded Demographic Table.
- Appendix F: Sigstore Components Mostly Used by our Sample Population.
- Appendix H: Summary of Factors Influencing Sigstore Adoption for Sigstore Users (Prior to Adoption).
- Appendix I: Summary of Factors Influencing Consideration of Sigstore Amongst Non-Sigstore Users.
- Appendix J: Additional Discussions & Implications of Results.
- Appendix K: Legible Image.

A. Traditional Signing Tools Workflow

The signature creation process begins when the maintainer's artifact is ready for submission. The signer generates a key pair that provides them with a private key for signing their software artifact and a public key for a verifier to validate the artifact's signature. Once the artifact is signed, it must be submitted to a package registry (such as PyPi, npm, or Maven) or the intended party. During this phase, the signer is responsible for making their public key available and discoverable so future users can verify the signer's identity. In the verification phase, the verifier retrieves the signer's public key, uses it to decrypt the signature file, and checks for equivalency with the software artifact's hashed digest. We show a typical traditional package signing workflow in Figure 5 compared to a next-generation workflow in Figure 6.

B. Interview Protocol

Table I gave a summary of the interview protocol. Here we describe the full protocol. We indicate the structured questions (asked of all users) and examples of follow-up questions posed in the semi-structured style. Given the nature of a semi-structured interview, the questions may not be asked exactly as written or in the same sequence but may be adjusted depending on the flow of the conversation.

We include Section B & C of the protocol for completeness, although they were not analyzed in this study.

C. Codebooks

Table VII describes our codebook, with code definition, and example quote tagged with that code.

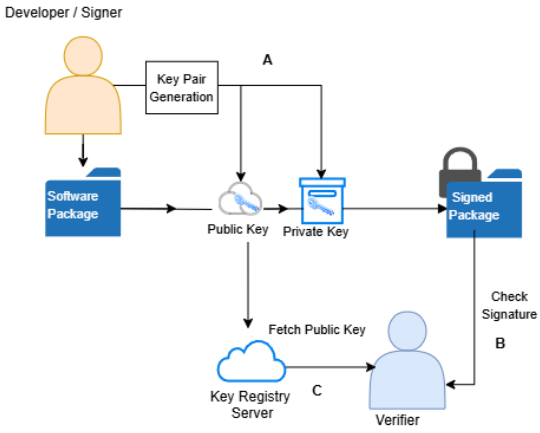

Fig. 5. Typical workflow for software signing and verifying signatures. The component author packages (A) and signs (B) their software. The signed package (C) and public key (D) are published. To use a package, a user downloads it (E) and its public key (F) and verifies the signature (G).

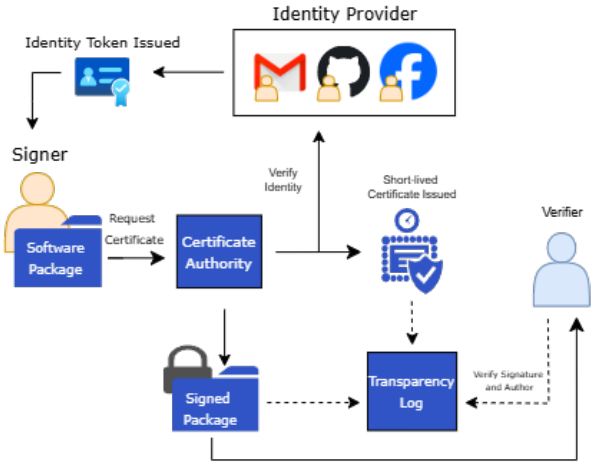

Fig. 6. Typical workflow for next-generation software signing. The component author requests a certificate from a certificate authority, which confirms the signer's identity through an identity provider. The signature and signer's identity are recorded in a transparency log, which the verifier can monitor to confirm the validity of the signature upon downloading the signed package.

D. Code Saturation

We present a saturation analysis of our studies in Figure 7. The green trendline (saturation curve) was measured by identifying the cumulative unique codes present in each interview. We observe saturation at the eleventh interview, indicating that no new unique codes were identified after participant 12. Additionally, the blue trendline (number of codes) shows that the lowest number of codes was obtained from subject S4. This is also reflected in the orange trend line (unique number of codes).

E. Survey Attempt

In addition to our interviews, we developed a survey distributed to Sigstore users at the 2024 SigstoreCon conference and through post-conference communications with the Sigstore user community. This aimed to gather perspectives from

TABLE VI

OUR FULL INTERVIEW PROTOCOL. THE TABLE ALSO INDICATES THE HISTORY OF THE PROTOCOL DEVELOPMENT: REMOVED (STRIKETHROUGH), CHANGED (⋆) AND RE-ARRANGED (†) QUESTIONS. WE ONLY ANALYZE PARTS A AND D IN THIS STUDY

Themes	Sub-themes	Questions
A: Demographic		**A-1:** *What best describes your role in your team? (Security engineer, Infrastructure, software engineer, etc)* **A-2:** *What is your seniority level? How many years of experience?* **A-3:** *What is the team size?* **A-4:** † What are the team's major software products/artifacts? ~~**A-5:** What type of Organization/company?~~
B: Software Supply Chain Failure Experienced		**B-1:** *Describe briefly the team's process from project conception to product release and maintenance – This is to understand the unique context of each practitioner's software production case.* **B-2:** ⋆ What do you consider a Software Supply chain attack/incident to be? **B-3:** Can you describe any specific software supply chain risks (or incidences with third-party dependencies, code contributors, open source, etc.) that your team has encountered during your software development process? • How were these addressed? (Alternatively – How did this affect the decision to implement software signing? **B-4:** †What are the team's major software products/artifacts? (Moved To Demographic from Prevalent SSC risk section After Pilot) **B-5:** What is your team's greatest source of software supply chain security risk? • Project components(Third-party dependencies, build process, code contributors, etc) • † Between the Software Engineering Process vs project components which constitute a greater source of risks?
C: Software Signing in Mitigating Software Supply Chain Risks		**C-1:** If no incident has been recorded (depending on the answer from Section B): why did the team choose to implement software signing? • Is software signing the team's major strategy to secure its supply chain? Any other complimentary security efforts and methods? • Are these strategies (software signing and complementary security techniques) influenced by regulations and standards? • What challenges or obstacles did your team face while integrating Software Signing into your supply chain security practices? • ⋆ How do different team members contribute to the implementation of Software Signing? What roles and responsibilities are involved? (After Practise). • Possible Follow up – Do you think Software Signing on its merit (if implemented right) is good enough to completely secure a supply chain? (Added after Pilot).
	Third-party dependencies	**C-2:** What are the team's peculiar selection strategies for Third-party dependencies? • How does the presence of a signature influence this decision? • How is the authenticity of this signature verified? • Any other methods/practices to ensure the trustworthiness of the third-party dependencies before integrating them into your projects? • Possible Follow-up – Can you describe these methods/practices? **C-3:** What influences the team to discontinue the use of a package? • How does the Signature on the dependency influence this decision? **C-4:** How does the team manage its third-party dependencies' security? • How do you maintain awareness of potential vulnerabilities or threats related to third-party components?
	First-party source code	**C-5:** Is software signing a requirement for team developers (insider threats) and open-source contributors (if any)? • how does the team use software signing to protect its source code (what parts of the process is signing required e.g. Commit signing)?
	Build Process	**C-6:** (How) Does the team utilize signing in its build process?
	Package Artifact/Deployment	**C-7:** How is Signing used to protect the following from compromise? • how Artifact's build binaries/deployment pipeline • Artifact's repository
		C-8: ⋆ How does the team evaluate the effectiveness of their Security processes/Signing Implementations? Any Feedback mechanism? (This question is derived from [93]).
D: Signing Tool Selection		**D-1:** *What software signing tooling does the team use (If not mentioned yet)* **D-2:** *What factors did the team consider before adopting this tooling over others?* **D-3:** *What was the team's previous signing practice before the introduction of current tooling?* **D-4:** How does your team implement this tooling (which components of tool does the team use)? **D-5:** Have you encountered any challenge(s) using this tool of choice? How have you coped with this limitation(s)? • Are there any areas where you believe further enhancements could be made in your current implementation of these methods? **D-6:** Have you/your team considered switching this tool for another? • What other tools have been considered or are currently being considered?

those who might not have had time for an interview. We exclude this data from our work due to several factors that compromise the quality of the results; these factors include respondents who completed the survey in an unrealistically short amount of time and those who failed to complete the entire survey.

Here, we summarize the answers to an excerpt of the questions responded to in our survey:

What signing tool did your team use prior to the introduction of Sigstore?: Among 7 participants who utilize Sigstore, 3/7 responded that they use PGP/GPG-based signing and 4/7 did not use any toolings prior to using Sigstore.

Which of the following factors have influenced your team's use of signing? (select all): Among 7 participants who utilize Sigstore, 4/7 selected "Organizational policies," 4/7 with "Best Practices/Other standards and recommendations" and 3/7 selected "Awareness of prominent SSC attack."

Which Sigstore components does your team use? (select all): Among 7 participants who utilize Sigstore, 3 responded to this question, and all 3 participants selected Cosign, Fulcio, and Rekor.

In which of the following ways have you implemented Sigstore?: Among 7 participants who utilize Sigstore, 3 responded to this question, with 2 responding that they use a

TABLE VII
EXCERPTS FROM THE FINAL CODEBOOK WE USED FOR OUR ANALYSIS.

Code	Definition	Sample Quote
Sigstore Criticisms	Criticisms and weaknesses associated with implementing Sigstore for signing.	S3: "We do have some concerns about the way Fulcio operates as its own certificate authority. So we've been looking at things like OpenPubkey as something that removes that intermediary and would allow you to do identity-based signing directly against the OIDC provider. "
Sigstore Improvements	Suggested improvements for Sigstore.	S9: "I would say probably integrations within different platforms, so say you use GitLab or GitHub or Jenkins or something like that. Try to leverage some type of integration to make that, I guess, less friction for the developers to implement, something like that."
Sigstore Strengths	Strengths of Sigstore and reasons why subjects prefer using Sigstore.	S7: "The convenience and the good user experience is probably what keeps us using Sigstore "
Sigstore Alternatives Considered	Alternatives of Sigstore that were or are being considered by the subject (for subjects whose primary signing implementation is Sigstore).	S5: "We are also looking at other new alternatives that are coming up, like OpenPubkey. Docker just released a new standard called OpenPubKey, which is trying to solve this issue where they will not require you to have infrastructures and stuff."
Sigstore Components Used	Components of Sigstore (Cosign, Fulcio, etc.) being used by the subject.	S2: "Mostly fulcio is what's integrated into our product."
Notary Criticism	Criticisms and weaknesses Associated with Notary – promoting users to stop using them.	"S5: "We were using Notary, and Notary only verifies that the image was signed using a specific key, but you cannot verify that who is the owner of the key."

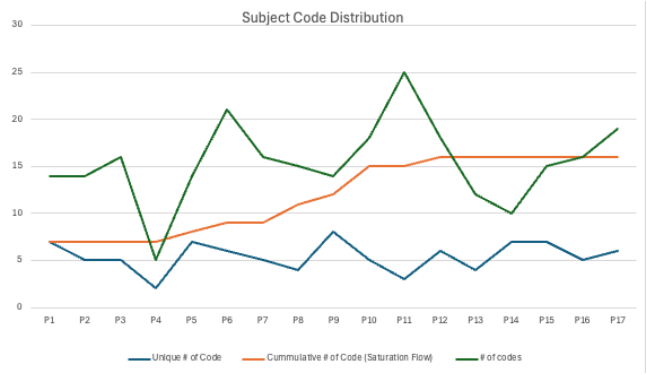

Fig. 7. Saturation curve. Interviews are plotted in the order in which they were conducted. Each interview covered many topics in detail (orange line, blue line). However, as the green line shows, our results saturated (*i.e.,* stopped observing new perspectives) around interview 11.

private deployment of Sigstore and 1 responding that they use a public instance of Sigstore.

Do you use or require Sigstore (signing) at any of the following stages of your product?: Among 7 participants who utilize Sigstore, 3 responded to this question. All 3 participants shared the "Build/Deployment/final product" stage as part of their answer.

How do you verify Sigstore signatures? (select all): Among 7 participants who utilize Sigstore, 3 responded to this question. All 3 participants shared selecting certificate Verification as part of their answer. 2/3 selected that they check

TABLE VIII
SIGSTORE FUNCTIONALITIES USED BY SUBJECTS.

Tool	# of Subjects
Cosign (Signing & Verification CLI)	9
Certificate Authority (Fulcio)	9
Keyless Signing & Identity Manager (OIDC)	9
Transparency Log (Rekor)	8
Gitsign (Commit Signing)	4
Component Customization & Local Sigstore Deployment	2

Rekor logs.

F. Sigstore Components Most Commonly Used by Our Sample Population

G. Expanded Demographics Table

H. Factors Influencing Sigstore Adoption (Sigstore Users)

I. Factors Influencing Sigstore Consideration (Non-Sigstore Users).

J. Additional Discussions & Implications of Results

In addition to the implications discussed in §VI, we outline further implications of our findings below.

1) Generalizability of Findings to Other Classes of Tools in Software Engineering: Our findings reinforce several adoption factors noted in prior literature on open source tools—such as ease of use, compatibility with existing technologies, and integration flexibility—which were especially salient for next-generation signing tools like Sigstore. They also highlight

TABLE IX

SUBJECT DEMOGRAPHICS. FOR ANONYMITY, WE USED GENERIC JOB ROLES, NOT SPECIFIC TITLES. "*Senior management*" REFERS TO SENIOR MANAGERS, DIRECTORS, AND EXECUTIVES; "*Technical leader*" TO SENIOR, LEAD, PARTNER, AND PRINCIPAL ENGINEERS; AND "*Engineer*" AND "*Manager*" TO MORE JUNIOR STAFF. WE HIGHLIGHT THE TYPE OF SOFTWARE SUBJECT'S IMMEDIATE TEAM PRODUCES. * REFERS TO CASES WHERE SUBJECT'S TEAM BUILT OTHER TYPES OF SOFTWARE IN ADDITION TO THE MAIN STATED. *O* REFERS TO CASES WHERE SUBJECT BELONGED TO MULTIPLE TEAMS WITH OTHER PRODUCTS THAN THE ONE STATED.

ID	Role	Experience	Software Type
P1	Research leader	5 years	Internal POC software
P2	Senior mgmt.	15 years	SAAS security tool*^{O}
P3	Senior mgmt.	13 years	SAAS security tool*^{O}
P4	Technical leader	20 years	Open-source tooling
P5	Engineer	2 years	Internal security tooling
P6	Technical leader	27 years	Internal security tooling*^{O}
P7	Manager	6 years	Security tooling*^{O}
P8	Technical leader	8 years	Internal security tooling*
P9	Engineer	2.5 years	SAAS security*
P10	Engineer	13 years	SAAS security*
P11	Technical leader	16 years	Firmware*
P12	Technical leader	4 years	Consultancy
S13	Senior mgmt.	16 years	Internal security tool
P14	Research leader	13 years	POC security Software*
P15	Senior mgmt.	15 years	Internal security tooling
P16	Senior mgmt.	15 years	SAAS
P17	Manager	11 years	Security tooling

TABLE X

REASONS PRACTITIONERS CHOOSE OR SWITCH TO SIGSTORE BEFORE ADOPTION. PRACTITIONERS' DECISIONS ARE DRIVEN PRIMARILY BY HUMAN/SOCIAL FACTORS—SUCH AS COMMUNITY CONTRIBUTION, FRUSTRATION WITH LEGACY TOOLS, AND USABILITY ISSUES—SUPPLEMENTED BY SIGSTORE'S TECHNOLOGICAL CAPABILITIES AND REINFORCED BY MACROENVIRONMENTAL PRESSURES LIKE REGULATIONS AND TRUST IN THE CNCF ECOSYSTEM.

Topics & Associated Examples	Subjects
Human/Social Factors	
Practitioners Contribute to Sigstore	6 – P2, P4, P6, P7, P8, P10
GPG Issues	6
Low adoption rates	P1, P10
Key management issues & Other usability concerns	P1, P6, P9, P12, P13
Steep learning curve	P9, P12
Compatibility with newer technology	P12
Notary Issues	2
Non-demand from customers	P9
Compatibility with other tools	P9
Lack of regular updates	P9
Key & Identity Management	P5
Proprietary Tool Issues	1
Difficult to setup	P11
Technological Factors	
Available Sigstore Functionalities	3 – P1, P5, P10
A transparency log, etc	P5, P10
Compatibility to other Tools	P1
Macroenvironmental Factors	
Regulation & Standards	4 – P5, P6, P11, P13
Large User Community	1 – P8
Inherent Trust of Creators	1
Trust of CNCF products	P3

TABLE XI

NON SIGSTORE USERS: TOOLING ISSUES CAUSING A CONSIDERATION OF SIGSTORE AND OTHER NEXT-GEN TOOLS.

Tool	Subjects	Topic
Internal tool	S8, S11	***Tooling Issues*** (1) Integration with other tools/platforms**T** – S8, S11 (2) Non unified tool implementation across teams **O** – S8, S11 (3) Signer ID Management **T** – S11 (4) Usage info/documentation unclear **T** – S11 (5) Verification not built in with tool **O** – S8 (6) Non transparency in signing infrastructure **T** – S11 (7) Security of Signing Infrastructure **T** – S11 (8) Engineers don't understand the current tool **P** – S11 ***Considered (or Adopted) Fix*** (1) Organization building new tools with Next-gen features – S8, S11
Notary-V1	S10	***Tooling Issues*** (1) Key Management**T** (2) Lack of an inbuilt signing policy **T** (3) Engineers don't understand the current tool **H** – S11 (4) Integration with other tools/platforms**T** ***Considered (or Adopted) Fix*** (1) Organization has adopted Openpubkey for key management (1) Organization considering Sigstore, and Notary V2
PGP	S13	(1) ***Tooling Issues*** (1) Key Management**T** (2) Non transparency in signing infrastructure **T** (3) Security of Signing Infrastructure **T** ***Considered (or Adopted) Fix*** (1) Use of Key discovery and delivery systems like TUF[4]

additional considerations that may generalize to other classes of secure software engineering tools example the role played by macroenvironmental factors like the tooling community, and regulations.

Furthermore, our results suggest adoption patterns not widely discussed in existing research. In particular, we observe the emergence of a migration pathway model, where teams incrementally adopt next-generation tools. Several non-Sigstore users reported experimenting with or partially integrating smaller solutions—such as OpenPubKey and TUF—to meet specific needs like key delivery and management. This suggests a stepwise transition process in tooling adoption that may be common across emerging tool ecosystems. This model can also be important to understand and further propose a streamlined organizational adoption pathway for software engineering tools.

2) The Surprising Role of Macroenvironmental Factors in Software Signing Tool Adoption: Many participants cited their involvement in the Sigstore community as a key reason for adopting the tool (**??**). While this may partly reflect our sampling, half of the non-Sigstore users who adopted external tools also emphasized community affiliation as a motivator.

This illustrates the influence of macroenvironmental factors—particularly community engagement—as captured in our usability framework. Challenges like setting up private Sigstore instances or accessing usage knowledge were frequently linked to the absence of sufficient community support (**??**). Other macro-level influences, such as regulatory pressure and trust in tool creators, also shaped adoption decisions.

These findings echo prior work on the importance of community in developer tool adoption [95]. Toolmakers should actively foster supportive, trustworthy communities around

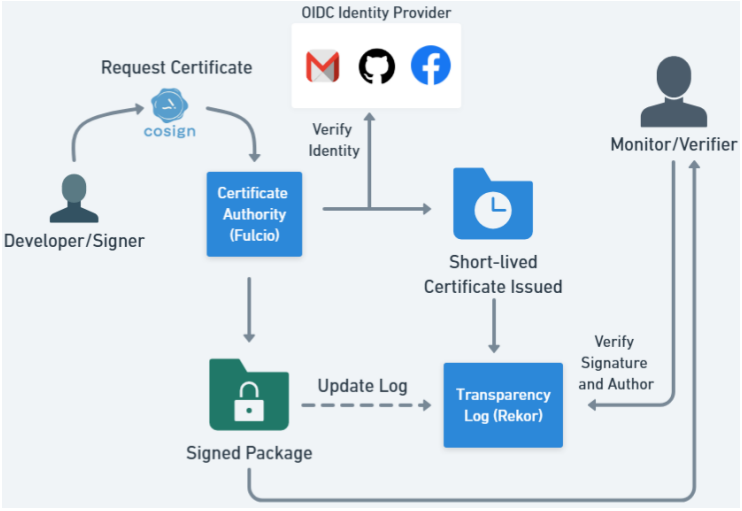

Fig. 8. Sigstore Signing Workflow. The software author requests a certificate from a certificate authority, which confirms the signer's identity through an identity provider. The signature and signer's identity are recorded in a transparency log, which the verifier can monitor to confirm the validity of the signature upon downloading the signed package.

their tools, as these networks can be pivotal to both adoption and long-term sustainability.

K. Legible Images

We have added a more legible version of our Sigstore workflow imageFigure 3 here Figure 8.