ARTIFACT EVALUATED
USENIX
AVAILABLE

Why Johnny Adopts Identity-Based Software Signing: A Usability Case Study of Sigstore

Kelechi G. Kalu
Purdue University
kalu@purdue.edu

Sofia Okorafor
Purdue University
sokorafo@purdue.edu

Tanmay Singla
Purdue University
singlat@purdue.edu

Sophie Chen
Carnegie Mellon University
scchen@andrew.cmu.edu

Santiago Torres-Arias
Purdue University
santiagotorres@purdue.edu

James C. Davis
Purdue University
davisjam@purdue.edu

Abstract

Software signing is the most robust method for ensuring the integrity and authenticity of components in a software supply chain. Legacy key-managed signing tools (*e.g.,* OpenPGP) burdened practitioners with key management and signer identification, creating both usability challenges and security risks. A new class of identity-based signing tools automate many of these concerns, but little is known about their usability and its effect on their adoption and effectiveness in practice. A usability evaluation can clarify the extent to which identity-based designs succeed and highlight priorities for improvement.

To fill this gap, we conducted the first usability study of Sigstore, a pioneering and widely adopted exemplar of identity-based signing. Through interviews with 17 industry experts, we examined (1) the problems and advantages associated with practitioners' tooling choices, (2) how and why their signing-tool usage has evolved over time, and (3) the contexts that cause usability concerns. Our findings illuminate the usability factors of identity-based signing tools and yield recommendations for toolmakers, adopting organizations, and the research community. Notably, components of identity-based tooling exhibit different levels of maturity and readiness for adoption, and integration flexibility is a common pain point but potentially mitigable through plugins and APIs. Our results will help identity-based signing toolmakers further strengthen software supply chain security.

1 Introduction

The reuse of software components in modern software production creates complex supply chains that are susceptible to the unauthorized introduction of code [6, 31, 84, 97]. To mitigate this risk, engineers need *provenance*—evidence of actor authenticity and artifact integrity [10]. Software signing provides the strongest guarantee of provenance currently available [88] and has thus been widely advocated by academia, industry consortia, and government regulators [13, 21, 32, 92].

Like other security practices, the effectiveness of software signing in practice depends on the availability and usability of tools that support it. Traditional signing tools such as OpenPGP are famously challenging to use [96], with several "Why Johnny Can't Encrypt"–inspired studies consistently revealing usability challenges that limit adoption [37, 72, 96]. In recent years, identity-based systems (*e.g.,* Sigstore [100], OpenPubKey [15], SignServer [74]) have been introduced that automate much of the process using short-lived keys and external identity providers [15]. However, despite growing adoption driven by recent supply chain attacks and efforts to lower usability barriers [28, 89, 100], we lack evidence about the usability of these identity-based tools and need to understand how that affects adoption in practice [60]. The innovations in identity-based approaches to signing motivate the need to evaluate how usability concerns affect their adoption and effectiveness. A usability evaluation can clarify the extent to which identity-based designs succeed, highlight how they compare to legacy key-managed signing approaches, and surface priorities for improvement.

To this end, we offer the first empirical study of the usability of identity-based software signing tools. Our work examines Sigstore, a pioneer identity-based signing tool that has seen rapid uptake across major open-source ecosystems [48, 58, 81], cloud-native projects, and corporations [76–78, 95], involving a large number of engineers worldwide. We interviewed 17 experienced security practitioners to investigate the usability concerns that drive or discourage Sigstore's adoption. We adopted an exploratory data analysis approach, because this study addresses a new class of tools that has not previously been analyzed. Our analysis centered on two aspects: practitioners' direct experiences with Sigstore and their perceptions of its usability relative to other signing tools.

Our study highlights the usability factors practitioners weigh when adopting identity-based software signing tools. We show that while identity-based signing tools ease legacy key-managed challenges, adoption is still constrained by integration hurdles, privacy concerns, and organizational factors. Second, our results provide adopting organizations with insight into the readiness levels of these tools, helping them manage adoption risks. Finally, we offer toolmakers and designers concrete directions for improving identity-based signing tools. Together, these findings provide formative feedback situated in real-world usage contexts.

In summary, we make the following contributions:

1. We report on the difficulties and advantages of using Sigstore, an identity-based software signing tool.
2. We discuss how practitioners' choices of software signing tools change over time and the factors influencing those decisions.
3. We discuss how organizational deployment contexts influence the usability concerns raised by practitioners.

Significance: Our work contributes to the security of software signing tools and, more broadly, to software supply chains. By unpacking the usability factors that promote and hinder Sigstore's adoption, we inform the ongoing refinement of its workflows and provide a template for evaluating and improving similar identity-based signing solutions. Because Sigstore exemplifies other identity-based software signing tools, our findings can be generalized to guide the design and implementation of this class of software, ultimately advancing usability (and thus security) across the software supply chain.

2 Background & Related Works

In this section, we review software signing technologies (§2.1) and usability assessments (§2.2).

2.1 Software Signing

Software signing ensures the authorship and integrity of a software artifact by linking a maintainer's cryptographic signature to their software artifact. From literature, the implementation of software signing is shaped by several factors: the software artifacts to be signed (*e.g.,* driver signing [14]), the root-of-trust design (*e.g.,* public key infrastructure/PKI [101]), and ecosystem policies (*e.g.,* PyPi [87] and Maven [80]).

2.1.1 Software Signing Tools

Software signing tools can be broadly grouped into two families [89]. *Legacy key-managed* tools such as GPG [1] follow a conventional public-key model, relying on long-lived asymmetric key pairs and delegating key-management responsibilities to the signer; trust is typically established through endorsements by other signers. In contrast, *identity-based* signing tools (described as "Next-generation signing" by Schorlemmer *et al.* [89]) 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 [100], OpenPubKey [15], and SignServer [74], reduces long-term key handling and often embeds transparency and auditability through append-only logs. We discuss the architecture and workflow of identity-based signing in §4.2 using Sigstore as a case study.

2.1.2 Tooling Effects on Signing Adoption

Previous studies [28, 55, 88] recognize that tooling influences whether and how software signing is adopted. For example, Schorlemmer *et al.* [88] perform a measurement study of several software package registries, identifying tooling as a key determinant of signature quality; their results indicate that "dedicated tools" lead to higher-quality software signatures. Kalu *et al.* [55] examine the implementation and adoption of signing in real-world organizational settings, highlighting tool usability as a central technical challenge for using software signing in practice. Their study focuses on how organizations incorporate signing into their existing software delivery processes, without centering on any particular tool. Although these works highlight tooling, and, in the case of Kalu *et al.*, usability, as important concerns, they do so only at a high level and provide little concrete guidance on what makes a signing tool usable or unusable in practice.

In this study, by contrast, we examine the specific usability of identity-based signing tooling, using Sigstore as a case study. We concretize formative usability concerns for Sigstore and extend Kalu *et al.'s* [55] work by specifying which features and design choices make identity-based signing tools like Sigstore usable or unusable in practice, and by translating these findings into concrete improvement guidance.

2.2 Usability Studies in Software Signing

2.2.1 Usability — Definition and Evaluation Approaches

Tools enable engineers to adopt new software development practices [85]. A usable tool is one whose functionality effectively supports its intended purpose, achieving product goals [60, 61]. Studies assessing implementation strategies of tools are typically framed as usability evaluations [3].

To evaluate a tool's usability, studies typically follow a *usability evaluation framework* [3]. There are two kinds of approaches. Summative work measures a tool's effectiveness [11, 52], *e.g.,* by task success rates, completion time, and user satisfaction. Formative assessments provide feedback to guide ongoing improvement [60, 93]. Given our aims, we took a **formative** approach. We considered several formative frameworks [57, 86, 90, 94] and found that Cresswell *et al.'s* four-factor framework [54] best explained our data. That framework is summarized in Figure 1.

Usability depends on the context in which a tool is used. Many factors, such as organizational objectives, personnel, policies, and compliance requirements, can make a tool usable in one setting and impractical in another [29, 96]. This contextual aspect informs our study design.

2.2.2 Empirical Studies of Signing Usability

The usability of legacy key-managed signing tools, often studied in the context of encryption, remains a significant challenge. Whitten and Tygar's seminal study, "*Why Johnny Can't Encrypt*" [96], found PGP 5.0's interface too complex for non-experts, a problem that persisted in PGP 9 [72], especially around key verification, transparency, and signing. Subsequent work [9, 17, 71] has underscored the public-key model's inherent complexity. Several works have subsequently examined the use and automation of signing in specific domains, such as social media conversations [23, 36, 38].

These prior usability studies of signing tools are limited in applicability to software signing. Much of these works have focused on communication contexts such as email and message encryption, where users have less technical expertise [27, 56] than software engineers securing their software supply chains. These studies also examine only legacy key-managed signing technologies. We contribute the first usability study of identity-based software signing tools.

3 Knowledge Gaps & Research Questions

Our literature review shows a gap in formative assessments of software signing tools. Furthermore, no study has examined identity-based signing tools, which are structurally and functionally different from legacy key-managed signing tools. We therefore ask:

RQ1: What usability strengths and weaknesses of identity-based software signing tools shape their adoption in practice?

RQ2: What factors drive practitioners to adopt, retain, or replace software signing tools over time?

4 Methodology

We investigated our research questions with a case study. Figure 1 summarizes our methodology. This section details our research design, including the study rationale (§4.1), case study context (§4.2), instrument design (§4.3), data collection (§4.4), and analysis procedures (§4.5).

Our Institutional Review Board (IRB) approved this work. The protocol number is Purdue-IRB-2023-841.

4.1 Rationale for Case Study

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 [34, 45, 70, 99], rather than a broad but superficial cross-analysis [67]. In selecting Sigstore, we adhered to the criteria outlined by Yin [99] and Crowe *et al.* [34]. Specifically, we considered the intrinsic value and uniqueness of the case study unit (high), access to data (adequate), and risks of participation (low). In §6 we discuss generalizability.

4.2 Case Study Context – Sigstore

Following guidelines outlined by Runeson & Höst [70], we describe the importance of our selected case study context and the technical details of the system.

4.2.1 Sigstore in Practice

Sigstore provides *keyless signing*, where developers authenticate using existing OIDC identities (e.g., GitHub, Google) instead of generating and distributing long-lived keys. It issues short-lived (ephemeral) certificates that simplify key management and reduce long-term security risks compared to legacy key-managed signing tools. Within 13 months of its public release in 2021, Sigstore recorded 46 million artifact signatures [48]. It has since gained broad adoption across open-source ecosystems (*e.g.,* PyPI [58], Maven [81], NPM [79]) and major companies (*e.g.,* Autodesk [78], Verizon [77], Yahoo [98]). Given its rapid uptake, integration into critical ecosystems, and pioneering design features, Sigstore provides a compelling case study for examining the usability of identity-based software signing tools.

4.2.2 Sigstore Components & Workflow

Following Newman *et al.* [100], we outline Sigstore's architecture, components, and workflow to orient the reader.

- **Cosign:** Command-line and library support for creating, verifying, and bundling signatures and attestations. Integrations may differ across ecosystems (*e.g.,* GitHub Actions vs. Maven plugins).
- **Identity Provider:** Services such as GitHub or Google that authenticate signers and issue short-lived identity tokens (*e.g.,* via OpenID Connect).
- **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:** Tools that validate the signer's identity, certificate status, and signature integrity.

Figure 2 summarizes the Sigstore software signing workflow. First, the signer authenticates with an OIDC provider to obtain a short-lived token. Next, the certificate authority

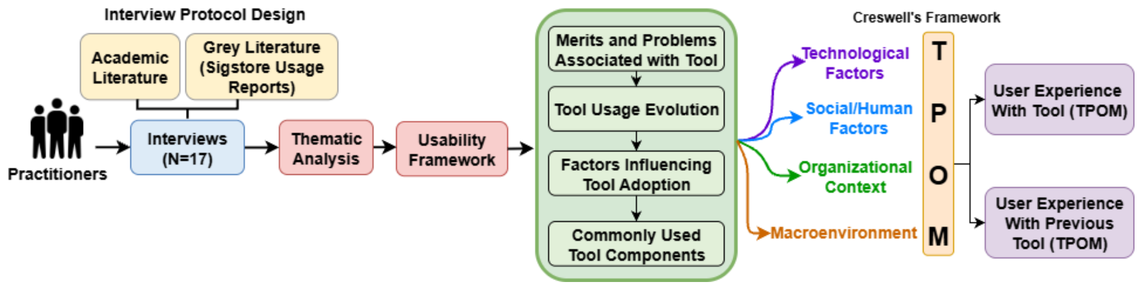


Figure 1: Study Methodology and Usability Framework. We developed our interview protocol by reviewing academic and grey literature on software signing, then conducted 17 practitioner interviews. Data were analyzed through thematic analysis and mapped onto Cresswell's formative usability evaluation framework [54]. The framework organizes usability concerns into four factors: Technological (T) — technical characteristics such as performance, ease of use, and data integrity; Social/Human (P) — user experience, and correct use of features; Organizational (O) — internal factors such as training, leadership, and resource support; and Macroenvironmental (M) — external influences including regulation, economics, and ecosystem community.

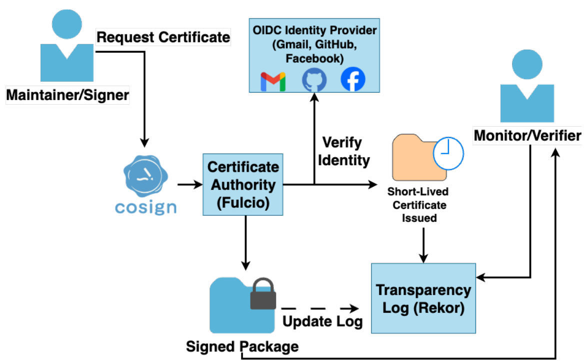


Figure 2: Sigstore Signing Workflow. The software author requests a certificate from a certificate authority, which confirms the signer's identity through an identity provider. The signature and signer's identity are recorded in a transparency log, which the verifier can monitor to confirm the validity of the signature upon downloading the signed package.

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.

4.2.3 Commonality with other Identity-Based Tools

Sigstore, like other identity-based signing tools such as OpenPubKey [15], AWS Signer [4], and SignServer [74], 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. We therefore use Sigstore as a focal case, while expecting many of our observations to transfer to similar identity-based signing tools.

4.3 Instrument Design & Development

Per §2.2, our study is a *formative usability* study. This type of evaluation offers methodological flexibility and prioritizes input from experienced practitioners to better articulate a system's usability concerns and produce actionable guidance [60]. To elicit long-form, context-rich opinions and detailed observations, we employed semi-structured interviews to examine various aspects of tooling usability (*e.g.,* implementation challenges, perceived benefits, and comparisons to other signing tools), following guidelines from Saldaña [73]. This approach allowed us to pose a consistent core of questions across all participants while probing unique aspects of each participant's circumstances.

Sigstore is an emerging (though maturing) technology, and case study research recommends exploratory designs in such contexts [70, 99]. To develop our interview protocol, we therefore pursued open, inductive inquiry to capture the full range of practitioner usage experiences. After this exploratory phase, we mapped our findings onto a formal usability framework, ensuring it reflected practitioners' experiences rather than constraining data collection to preconceived categories (§4.5). Therefore, in constructing the protocol, we:

1. Reviewed grey literature, Sigstore blogs [75] and usage reports [7, 49, 76–78, 91, 95], to ground our questions in real-world implementations and updates.

2. Used a snowball review of top papers on signing primitives [28, 100] and relevant empirical studies (see §2) to identify gaps and avoid redundant questions. We seeded the review with works from prominent cybersecurity and software engineering venues (*e.g.,* USENIX Security, IEEE S&P, ICSE, FSE) published in the past 10 years.

Table 1: Summary of interview protocol. See §9 for the full protocol.

Topic	Sample Questions
A. Demographics	What is your role in your team?
B. Contextual Information (Risks & Use of Signing)	Can you describe any specific software supply chain attacks (*e.g.,* incidents with 3rd-party dependencies, code contributors, OSS) your team has encountered? How does the team use software signing to protect its source code (what parts of the process is signing required)?
C. Signing Tool Usability Questions	
C1.	What factors did the team consider before adopting this tool/method (Sigstore) over others?
C2.	What was the team's previous signing practice/tool before the introduction of Sigstore?
C3.	How does your team implement Sigstore (which components of Sigstore does the team mostly use)?
C4.	Have you encountered any challenge(s) using this tool of choice?
C5.	Have you/your team considered switching Sigstore for another tool?

Table 2: 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. We add interview duration to an extended version of this table (Appendix C).Table 6 gives organizational info.

ID	Role	Experience	Software Type
P1	Research leader	5 years	Internal POC software [4]
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
P13	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

Next, we refined our interview protocols through a series of initial interviews [12]. We first conducted practice interviews with the secondary authors of this work, followed by pilot interviews with the first two interview subjects. We modified our protocol after practice and pilot, affecting 8 questions.[1].

Table 1 summarizes our final interview protocol[2]. After the demographic section (Topic A), our interview instrument consisted of two main topics (B & C). **Topic B** comprised contextual questions about the supply chain risks that prompt signing and how signing is implemented across the software production process (*e.g.,* where signing is required vs. optional, which artifacts are signed, and relevant workflow/policy constraints). **Topic C** presented questions to assess the usability of the signing tool(s) used by the participant's organization, including perceived strengths and weaknesses, adoption rationale, implementation details, challenges, and any considerations about switching tools.

4.4 Data Collection

Population: Our study investigates signing tools using a case study of the Sigstore tool ecosystem. Accordingly, our participant pool focused on current and prospective users of this tool. Furthermore, since our research questions pertain to organizational decision-making, we targeted expert 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 [5]. 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 [39, 51] Steering Committee (ITSC). This initial invitation yielded 6 participants. We then expanded our recruitment through snowball sampling, relying on recommendations from initial participants (4) and authors' contacts (7) to recruit an additional 11 participants. We report on data from 17 participants.[3]

Participant Demographics: Our subjects were experienced security practitioners responsible for initiating or implementing their organization's security controls or compliance. They thus had 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 2. 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 of Sigstore by non-users.

Interviews: We conducted our interviews over Zoom. Each interview lasted ∼50 minutes. The lead author conducted these interviews. We offered each subject a $100 gift card as an incentive in recognition of their expertise.

Survey Attempt: We recognize the importance of data triangulation in case studies and therefore fielded a follow-on survey to complement our interviews. We deployed a survey to validate and model the prevalence and stability of our in-

[1]Protocol revisions are summarized in the technical report appendix; see Appendix C.

[2]Questions in the protocol reflect our screening criteria (expert users of signing infrastructure), with later questions building on earlier answers.

[3]We conducted 18 interviews. Participant 18 was a senior security expert. While aware of the organization's overall use of signing, they reported limited knowledge of the signing tool's specific operation and deployment.

[4]POC: Proof of concept software: early-stage software that may be productized.

terview themes over time and across contexts. We tested our survey instrument in two practice runs with two members of our research team. Despite five months of repeated solicitation in the Sigstore community, we received only 13 completed responses, of which 7 were sufficiently complete for analysis (see §9). This regrettably echoes the well-documented difficulty of recruiting participants from highly specialized populations [83].

Data triangulation is encouraged but not mandatory in case study research. As Stake notes, combining multiple data sources strengthens the substantiation of constructs and hypotheses, yet practical constraints necessarily shape feasible designs [82]. We encountered such constraints.

4.5 Data Analysis

We analyzed data in two stages: an interpretive–exploratory thematic analysis, and then a retrospective mapping of emergent themes onto a formative usability framework.

First, we adopted an *interpretive* and *exploratory* [35] approach for initial analysis, as recommended for studies characterized by substantial uncertainty [69]. In our case, this uncertainty concerned how organizations adopt software signing as a security method, a decision that varies with organizational policies, security expectations, perceived threats, and customer requirements. This orientation motivated the use of a reflexive thematic analysis lens [8]. At the same time, we aimed to generate concrete, usable guidance for tool designers. To increase the systematicity and transparency of our process, we therefore incorporated selected coding-reliability practices, structured early coding, and agreement checks [66]. In short, reflexivity grounds our insights, and these reliability practices help make them more communicable and actionable.

Second, because our work is a usability study, we situated the exploratory findings in a usability context by applying a formative usability framework retrospectively to our inductively generated results. This retrospective mapping of emergent themes onto existing constructs is an established interpretive strategy in qualitative analysis. This method uses prior constructs to structure and extend inductively derived findings while preserving empirical grounding (*e.g.,* [19, 30, 42]).

Memoing & Codebook Creation. Two analysts participated in this analysis stage. After transcription[5] and anonymization, we began with data familiarization [24]. During Familiarization, the main analyst[6] read all interviews, and the secondary analyst read three transcripts to calibrate.

Next, the two analysts independently memoed a subset of 6 anonymized transcripts through three rounds of review (2 transcripts per round) and discussion, following the recommendations of O'Connor & Joffe [66] and Campbell *et al.* [53]. This process improved reliability of the analysis while making efficient use of resources.[7] After each round, we discussed emerging codes in meetings, which produced 506 coded memos in total. We distilled these memos into a codebook (36 categories) over several multi-hour meetings.

Next, to assess the reliability of the coding process, we combined the techniques outlined by Maxam & Davis [50] and Campbell *et al.* [53]. Specifically, we randomly selected an unanalyzed transcript, which we coded independently. Following Feng *et al.'s* [41] recommendation, we used the percentage agreement measure[8] to evaluate consistency, as both analysts had contributed to developing the codebook. We obtained an 89% agreement score, suggesting a stable coding scheme [53, 66]. We resolved disagreements by discussion.

Codebook Revision. Following the reliability assessment of our codebook and given the substantive agreement recorded, the main analyst proceeded to code the remaining transcripts in two batches of six and five transcripts per round, respectively. The lead analysts revisited earlier coded transcripts throughout. 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 five 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.* [44], we measured the cumulative number of new codes introduced in each interview (Figure 6). We saw saturation after interview #12. Each interview yielded a median of 15 codes.

Thematic Analysis & Usability Analysis Next, following Braun & Clarke [8], we derived initial themes from our code categories (see Table 1 for protocol focus). This reflexive thematic analysis provided an inductive account of participants' experiences and practices. The lead analyst generated the themes and then discussed and refined them with feedback from the secondary analyst.

From the emergent themes, we interpreted our findings using a formative usability framework. Lacking formative usability studies set in similar contexts in the Software Engineering literature, we followed advice [22, 40, 43] to seek theory from adjacent disciplines. Surveying candidate frameworks

[5] Interview recordings were transcribed by www.rev.com using human transcription to address spoken accents.

[6] The lead author is the main analyst in this study.

[7] Although there is no consensus on the ideal proportion of data to use, O'Connor and Joffe note that selecting 10–25% is typical [66]. We randomly selected six transcripts (∼35% of our dataset) to develop our initial codebook.

[8] Calculated as:

(matching code assignments / total code assignments) × *100*

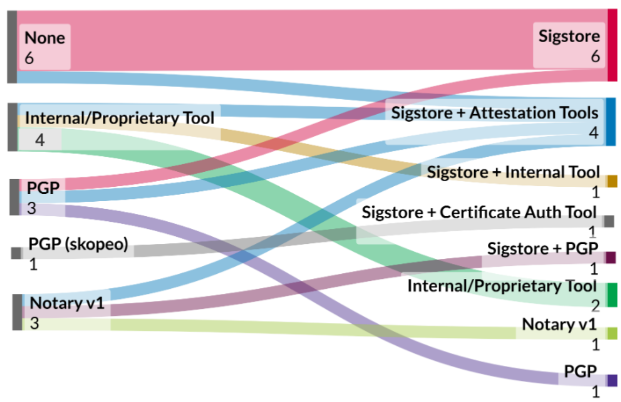


Figure 3: Self-Reported Evolution of Software Signing Tools. Sankey diagram showing participants' self-reported transitions between previously used and current signing tools in their organizations. Flows (e.g., from no signing to Sigstore) reflect participant accounts, not our observations.

beyond SE, we found three groups: formative usability assessments of software systems [86]; applications of technology systems in other disciplines [54, 57, 94]; and general usability frameworks [90]. Among these, we selected Cresswell *et al.'s* four-factor TPOM framework (*Technology, People, Organization, Macro-environment*) [54] since it best aligned with our inductively generated themes (see Figure 1). Our intent was to synthesize those themes into higher-level conceptual groupings that correspond to established usability determinants of adoption. The TPOM categories also suggest interventions, *e.g.,* People issues may not be resolved via Technology.

We then conducted a framework-informed analysis, mapping the emergent themes to the TPOM factors to refine and structure our interpretations. The lead analyst mapped our earlier generated themes and then discussed and refined the mappings with feedback from the secondary analyst.

5 Results

We begin by presenting a Sankey diagram (Figure 3) that illustrates the distribution of our participants between Sigstore and non-Sigstore users, and how their teams changed signing tool usage over time. Many participants (6) transitioned from using no tools to Sigstore. Others transitioned from existing tools to Sigstore: 3 from implementations that had some combinations of GPG/PGP (of which one practitioner reported combining PGP with Skopeo [68]), 2 from Notary, and 2 from proprietary or internal tools. 4 subjects did *not* adopt Sigstore.

To understand the usability concerns that Sigstore adopters face; and why practitioners did (and did not) change tools, we structure the results by RQ using Cresswell's framework.

Table 3: 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 subjects*
1. Signing Workflow & Verification	P1, P2, P7, P9, P14-17
2. Setting up with automated CI/CD actions	P9
3. No key distribution problems	P1
Use of Short-lived Keys & Certificate	*3 subjects* P2, P3, P15
Signer ID Management	*4 subjects*
1. Use of OIDC(Keyless) to authenticate signers	P3, P5, P10, P12
Compatibility with Several New Technologies	*4 subjects*
1. Integrability with SLSA build	P16
2. Integrability with several container registries/technologies	P12, P14
3. Integrability with several cloud-native applications	P15
Presence of a Transparency Log	*3 subjects*
1. Transparency logs increase security	P5, P14
2. Evaluation of signing adoption using logs	P9
s *Bundling Signatures With Provenance Attestations*	*2 subjects* P3, P4
Reliability of Service	*1 subject* P7
Macroenvironmental Factors	
Free/Open-Source	*2 subjects* P7, P17

5.1 RQ1: Strengths & Weaknesses of Sigstore

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

5.1.1 Sigstore Strengths

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 as summarized in Table 3. When mapped to the Cresswell framework, these factors fell into two categories: *Technological* and *Macroenvironmental*.

Technological Factors We identified five factors:

Ease of use: This was the most frequently discussed advantage of Sigstore over other tools. Participants commonly highlighted the simplicity of the Sigstore workflow. As P7 *(security tools, cloud security)* put it, *"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 ease-of-use advantages mentioned were the seamless integration of the Sigstore workflow into CI/CD automation using GitHub Actions and the reduced complexity of key management.

Identity management for Signers: Sigstore's integration of OIDC identity management was a strength per our participants. In legacy key-managed 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 streamlined the signing process by reducing the time spent distributing keys to key servers and searching for keys. This aligns with the intuition that identity management is easier than key management for developers [100]. 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... Sigstore is the easy choice because it just works on any registry, and integrating it to Kubernetes is easy."*

Use of Short-lived Keys/Certificates: Participants appreciated Sigstore's enforcement of short-lived keys and certificates. Some signing tools require users to manually set expiration times for cryptographic elements such as keys and certificates at the time of generation. That customizability also introduces the risk of long-lived cryptographic materials, increasing the likelihood of compromise. Participants preferred Sigstore's automated management of key and certificate lifetimes. As P2 *(SAAS, SSC security tool)* stated, *"PGP, you still have to figure out the key distribution problem....Sigstore uses short-lived keys, that solves a lot of those issues."*

Presence of a Transparency Log: Adopters valued the ability to audit signing actions. Sigstore's transparency log provides a tamper-resistant record of all changes to an artifact's metadata, a functionality not offered by other signing systems.

> **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."*

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."*

This enthusiasm reflects both the appeal of free/open-source (FOSS) software and the practical benefits of public accessibility. However, note that open-source $\neq$ free; enterprises may pay for private deployments, support, and integration.

Table 4: Summary of Practitioner-Reported Difficulties Using Sigstore. Difficulties are grouped by Cresswell factors.

Topics & Associated Examples	**Subjects**
Technological Factors	
Transparency Log Constraints	*6 subjects*
1. Not suitable for private setup	P2, P3, P6, P14, P17
2. Use in air-gapped/offline conditions	P2, P3, P9
Performance & Rate Limits	*6 subjects*
1. Rate limiting problems	P3, P7, P14, P15
2. Latency concerns	P6
Integrations & Tooling	*3 subjects*
1. GitLab/Jenkins	P9
2. Other unsupported technologies	P16
3. Attestation storage	P1
Documentation (Technical Aspects)	*6 subjects*
1. Private instance setup documentation	P9, P12, P15
2. Other documentation / usage issues	P1, P14, P17
Private Instance: Infra & Cost	*2 subjects*
1. Infra requirements & maintenance cost	P5, P6
Fulcio / Timestamping Workflow	*1 subject*
1. Timestamping issues	P3
2. Fulcio–OIDC workflow	P3
Software Libraries	*1 subject*
1. Unsupported software libraries	P7
Social / Human Factors	
Log Monitoring Burden	*1 subject*
1. Effort to monitor logs	P2
Support & Maintenance	*3 subjects*
1. Lack of dedicated support & maintenance	P15, P2, P6
Organizational Factors	
Regulatory Suitability (Cross-factor)	*1 subject*
1. Not suited for regulated organizations — ***M & O***	P3
Macroenvironmental Factors	
Community Maturity/Support	*1 subject*
1. Limited community support	P15

5.1.2 Sigstore's Weaknesses

We summarize the difficulties reported by participants in using Sigstore in Table 4. As we detail, most of these weaknesses mapped to multiple usability factors.

Macroenvironmental & Organizational The primary weakness along this dimension related to Sigstore's use in large enterprise contexts.

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.

> **P7** ***(security tools, cloud 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."

Technological Factors. We identified seven technological issues.

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 [64] 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 requirements 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 premises."*

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 time-stamping. 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.

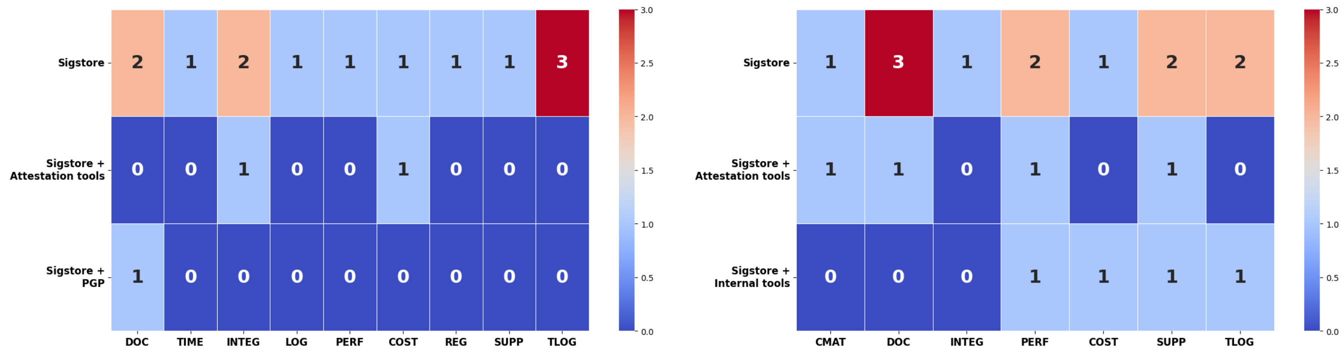
Software Libraries: The programming libraries of Sigstore were also reported as a challenge, particularly when participants sought to embed signing capabilities directly into applications. They noted the absence of well-supported, easy-to-use client libraries, which forced them to rely on workarounds such as shelling out to the CLI.

5.1.3 Impact of Issues on Sigstore Component Use

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. Only two successfully deployed private instances, likely due to these challenges (Table 4, §5.1.2). 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. Although participants shared experiences with primary signing workflow



(a) Subjects from small organizations (6 subjects). *(b) Subjects from large organizations (7 subjects).*

Figure 4: Tools vs. Evaluation Criteria. Reported usability issues across organizational contexts. Heatmaps show the number of unique participants raising each issue; darker cells indicate higher concentrations of reports. **Abbreviations:** DOC = Documentation (technical); TIME = Fulcio / Timestamping Workflow; INTEG = Integrations & Tooling; LOG = Log Monitoring Burden; PERF = Performance & Rate Limits; COST = Private Instance: Infrastructure & Cost; REG = Regulatory Suitability (Cross-factor); SUPP = Support & Maintenance; TLOG = Transparency Log Constraints; CMAT = Community Maturity/Support.

components, discussion of other ecosystem elements was notably absent. In particular, there was very little mention of log witnessing [47] or monitoring [2] for either Fulcio or Rekor. This is notable because transparency solutions rely on these parties to help ensure log integrity and detect log misbehavior. The lack of discussion may suggest that practitioners are either not yet engaging with these components or may not fully recognize their importance in the correct use of the ecosystem. We believe future work should examine gaps in user understanding and develop concrete user stories and guidance to help practitioners verify witnesses and use monitoring information effectively.

5.1.4 Effects of Organization and Signing Tools on Identified Usability Concerns

To provide additional context for the usability issues identified above, we highlight three observations from our analysis of organizational size and participant roles. See Figure 4 for visualizations.

Large and small organizations share similar pain points.: Across both large and small organizations, the densest cells occur for topics related to *private/air-gapped deployments*, *transparency-log constraints*, and *documentation for customized/ private instances*. In large organizations, technical leaders frequently surface additional concerns around *support and maintenance expectations*. In contrast, smaller organizations are more willing to combine Sigstore with other tooling to compensate for gaps. Notably, large organizations in our sample were more likely to retain internal/proprietary solutions or delay full Sigstore adoption. A usability lesson here is to prioritize private-deployment guidance (including monitoring playbooks), clarify support boundaries, and provide migration patterns and user stories for mixed stacks.

Scale effects: rate limits and latency concentrate in larger orgs: Performance and rate-limit issues are most prominent in medium/large organizations, consistent with higher CI/CD concurrency. This suggests the need for prescriptive CI recipes (e.g., batching, caching, retries/backoff, and concurrency guidance) alongside clearer Fulcio/OIDC and timestamping failure-mode documentation.

Medium organizations tend to "work around" with plain Sigstore: Participants from medium-sized organizations more often stick with plain Sigstore and work around gaps rather than self-host. This group benefits most from concise integration blueprints (GitHub/GitLab/Jenkins), ready-to-use attestation storage patterns, and targeted documentation updates, rather than heavy private-infrastructure guidance.

5.2 RQ2: Why Practitioners Change Tools

5.2.1 Drivers of Adoption Among 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. See Appendix C for the full summary of factors.

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."* While this pattern may partly reflect our interview sample, it highlights how prior involvement and community ties can strongly shape adoption decisions—a dynamic that could also apply more broadly in open-source ecosystems.

Available Sigstore Functionalities: Sigstore's functionalities also influenced its adoption. Several participants were drawn to the transparency log (as highlighted in §5.1.1).

> **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 (*e.g.,* EO-14028) played a 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 P14 cited GPG's smaller user base as a drawback. In addition, some participants expressed trust in industry consortia such as the CNCF (Cloud Native Computing Foundation), noting that Sigstore's development and governance under CNCF gave it additional legitimacy and credibility compared to tools maintained by smaller, independent groups.

> **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 with previously existing tools played a major role in practitioners' decisions to switch to Sigstore. Both GPG- and Notary-based signing implementations were reported as problematic. GPG has long been criticized for key management challenges and usability concerns, and 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.

5.2.2 Considerations from Non-Sigstore Users

In this section, we highlight reasons mentioned by practitioners that have stopped them (or their organization) from switching to identity-based tools like Sigstore (see Table 5). We also report issues experienced by these non-Sigstore users that have prompted a consideration (or in some cases partial adoption) of these identity-based tools.

Table 5: 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	P8, P11	(1) Third-party security risk from tool **T** – P11, P8 (2) Organizational use cases **O** – P8 (3) Centralizing organization's code signing **O** – P8 (4) Organization's open source policy **O** – P8
Notary-V1	P10	(1) Tool owned by organization **O** – P10
PGP	P13	(1) Sigstore privacy concerns **T** – P13 (2) Trust in Sigstore's maintainers **M** – P13

Barriers to Considering Sigstore: As highlighted in Table 5, most reasons for non-adoption were organizational.

> **P8** ***(internal security tool & cloud APIs, internet services)****: "When it comes to open source projects and open source stuff, our organization is skeptical. Policy and everything drives teams away...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. We do evaluate these things, but the scrutiny against code written outside of the company is extreme, and with good reason given the [supply chain] context."*

Other notable issues arose from technical concerns, like privacy risks associated with Sigstore's public instance and Macro environmental concern of 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 problems with existing tools as their main driver to alternatives, *e.g.,* incorporating

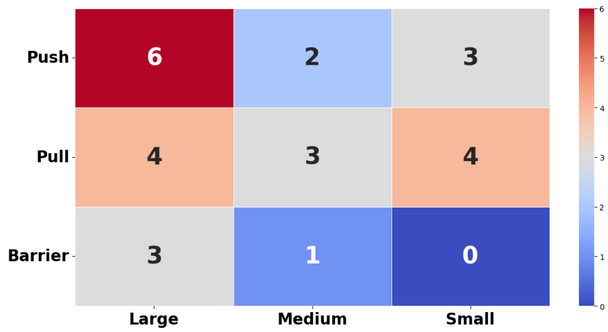


Figure 5: Push-pull-barrier heatmap.

identity-based features into internal tools or partially adopting new solutions. Although some subjects reported drawing design inspiration from identity-based workflows, none specifically mentioned Sigstore. P11 *(firmware & testing software, social technology)* states, *"I'm sure ... [the internal signing tool] was influenced by [an open-source identity-based tool]...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 Appendix C.

5.2.3 Effects of Organizational Context

Combining the results in §5.2.1 and §5.2.2 (Table 5) with organizational size, former signing tools, and current signing tools reveals several patterns. We summarize these in the Push-pull-Barrier heatmap (Figure 5).

Push, Pull, and Barrier Factors Influencing Adoption. We grouped issues raised by participants into *Push*, *Pull*, and *Barrier* factors. We define these factors drawing on Bansal *et al.'s* Push-Pull-Mooring framework for consumer switching behavior [25]. *Push* factors are problems with prior tools that drove current users toward Sigstore. *Pull* factors are attractions of the new tool (and ecosystem). *Barriers* are reasons that stalled or prevented adoption.

While large organizations exhibit the greatest number of barriers, they report push and pull factors in roughly similar measure. By current-tool status, many push factors come from teams now using Sigstore alongside other mechanisms (e.g., attestations, internal tooling, PGP), which often reflects phased introduction of Sigstore into existing processes (and vice versa). A summary of the coded factors appears in Appendix C. A usability takeaway is that switching is *primarily* driven by human/social push (e.g., frustration with legacy workflows) and only *secondarily* by pull from Sigstore features and regulatory pressure (Sigstore being referenced by some standards). Thus, even as technical innovations "pull" adoption, community-building and other social supports also materially "push" change.

Non-Adoption Reasons Are Often Organizational and Non-Technological. Non-adoption reasons are often organizational and non-technological. Although non-adopters are both pushed by shortcomings in internal tools and pulled by identity-based capabilities, adoption frequently stalls due to organizational constraints such as risk posture, support expectations, or regulatory requirements. This pattern is not unique to software signing—similar dynamics have been observed in other infrastructure domains (e.g., cloud technologies [16])—but it is particularly salient here because identity-based signing tools must be trusted across organizational boundaries. Our interpretation is therefore not that identity-based tools have failed, but rather that their technical attractiveness has created demand for more mature deployment models: private instances, enterprise-grade support, and stronger service guarantees.

Tool Migration Patterns. In our sample we observe recurring migration paths: *PGP* → Sigstore (sometimes retaining PGP during transition); *Notary v1* → Sigstore or *Notary v2*; and *internal/proprietary* → Sigstore (or hybrid). Organizations mostly transition between tools slowly. Larger organizations are more likely to prefer internal/hybrid arrangements (reflecting policy, privacy, and support concerns), whereas small/medium organizations are more likely to adopt "pure" Sigstore sooner, using hybrids as transitional steps or to meet offline/private constraints.

6 Discussion

We highlight the main takeaways from our results. This research is a case study of identity-based software signing tools (described in §2.1) using Sigstore. We discuss implications for this class of tools along three axes: a technological readiness assessment, their transformative potential, and the implications of the observed usability patterns.

6.1 TRLs and Identity-Based Signing Tools

Because Sigstore, our exemplar of identity-based signing, is relatively recent, the maturity of its core capabilities is uneven. Understanding this maturity helps organizations assess implementation risk and prioritize investment [62]. Our usability study provides a preliminary signal. Drawing on participant reports and interpreting them through the Technology Readiness Level (TRL) lens [62], we note that TRL is a technology-maturation framework (not a usability scale). In practice, lower TRLs (concepts, prototypes) tend to surface mostly *technical* concerns, while higher TRLs (operational environments) bring *non-technical* constraints (organizational fit, governance, support/SLAs, regulation) to the foreground. Our results fit this progression: *technological* factors were most frequently discussed. Several capabilities, *e.g.,* short-lived (ephemeral) keys, OIDC-based signer identity, and GitHub Actions workflows, received consistently positive mentions across different organizational contexts, suggesting higher maturity for these components (and analogous features in similar identity-based tools). In contrast, capabilities such as private/self-hosted deployments and operation in regulated environments surfaced moderate friction, indicating mid-level readiness and a need for stronger documentation, reference architectures, and support pathways. For organizations that have not yet adopted identity-based tools, these findings suggest comparatively lower risk for the first set of capabilities, while private/regulated deployments warrant careful evaluation and piloting. These assessments are preliminary and should be read in light of our case-study scope.

From Sigstore to the Broader Class: Sigstore demonstrates that identity-based signing can work at scale (across multiple organization sizes and use contexts in our sample), serving as a proof-of-concept for the class. Where we observed Sigstore-specific friction (e.g., private Fulcio/Rekor deployments, public-log privacy trade-offs), the underlying *class* implications are clear: enterprise deployment models, integrations, and operational guardrails are the next maturity hurdles *for identity-based signing generally*.

6.2 Transformative Potential

Sigstore, as the most widely deployed exemplar of identity-based signing, goes well beyond the minimal OIDC–key binding model offered by alternatives like OpenPubKey. In addition to identity-based signing and short-lived keys, Sigstore integrates a transparency log (Rekor) and broad ecosystem compatibility with attestations, container registries, and CI/CD workflows. Participants repeatedly emphasized these *additional* capabilities as unique differentiators, even while reporting usability frictions.

This dual reality—strong adoption coupled with persistent friction—underscores Sigstore's role as a living testbed for both practice and research. Sigstore is already being leveraged in production ecosystems (*e.g.,* Maven, PyPI, Kubernetes) and has catalyzed new research trajectories, such as privacy-preserving transparency [28] and diversified identity provisioning [20]. The broader implication is that Sigstore is not merely a tool to be adopted, but an infrastructure on which future signing solutions and empirical studies can (and are) being built. Widespread adoption makes its usability challenges consequential — and resolving them has cascading benefits across the supply chain. Its design decisions and shortcomings provide researchers with concrete leverage points for advancing the next generation of signing tools.

6.3 Usability Lessons for ID-Based Signing

While our results reveal different adoption drivers between Sigstore users and non-users, there exists some shared usability concerns—between Sigstore users and non-users (§5.2). Both groups reported similar technological issues, particularly around integration with other systems and the transparency of signing infrastructures. This suggests cross-cutting concerns that affect not only Sigstore users but also non-users. A preliminary look at enhancement/issue trackers for other identity-based signing tools (*e.g.,* OpenPubkey) shows similar themes [65]. Building on these observations, we distill actionable lessons for identity-based tools and outline opportunities for future work.

Automated Cross-Platform Integrations: To fully leverage flexibility, practitioners need maintained integration modules for popular CI/CD systems, registries, and orchestration platforms. Missing integrations and sparse documentation force teams to build custom connectors, increasing misconfiguration risk and undermining security guarantees.

Need for Inter-Platform Standards: Standardization of information exchange is a typical problem for new infrastructure (*e.g.,* SBOM [18]). Identity-based signing would benefit from uniform schemas for transparency events and identity assertions. Recent efforts like DiVerify [20], which harmonizes identity provisioning across multiple OIDC providers, take a step toward smoother data flows between signing components, transparency services, and external toolchains.

Verification Workflow: A core supply-chain property is transparency of actors and artifacts, which requires reliable, usable verification of signer identities [10]. Prior work reports verification pain in both organizational and open-source settings [55, 89]. Although Sigstore provides multiple verification paths (Cosign CLI, transparency log), participants reported log-related usability issues and limited automation elsewhere. Further automating verification would better realize the security benefits of this tool class.

Privacy Concerns: Privacy concerns emerged prominently around Sigstore's Rekor transparency log: five of the six participants who reported concerns with Rekor cited privacy as a significant issue. Similar concerns also surfaced for keyless signing via OIDC. These are not unique to Sigstore [28] but reflect broader tensions in cryptography between transparency and confidentiality. This underscores the need for privacy-preserving designs (e.g., selective disclosure, redaction regimes, witness policies) that balance auditability with confidentiality. In Creswell's terms, these concerns span macroenvironmental (policy/regulation), organizational (data governance and risk posture), and human/social (developer risk perception) factors, reinforcing that privacy is not solely a technical hurdle but a cross-factor adoption barrier.

Usability as a Limit on Realized Security: For security tools, usability problems are barriers to realizing effective security [46]. When tools are hard to use, their features may be partially deployed, misconfigured, or disabled [46]. Our findings show this in the context of identity-based signing: for example, rekor transparency-log usability concerns mean that some corporate users avoid the log feature, despite Sigstore supporting more privacy-preserving setups (which seem to be unusable in their context). As a result, these users miss the full auditability the log can provide. For example, P3 *(SAAS, SSC security tool)* said, *"Well, I think there's definitely reasons we haven't adopted some things like Rekor, and the biggest reason for that is being able to validate everything offline in our capped(air-gapped) environments. And so there's not a whole lot of value for us in having something in the transparency log if we can't prove somewhere else without access to Rekor, that it's in the transparency log."*

Population Scope & Future Populations: Our study primarily sampled experienced practitioners operating within organizations, which shapes both the opportunities and frictions we observed. Usability needs likely differ for other communities and skill levels—*e.g.,* individual open-source maintainers, small/startup teams, and security novices. In light of the biases from our particular sample, we recommend further research that (1) uses organizational contexts and user personas as a primary selection criterion; and (2) considers distinct signing use cases for a more fine-grained understanding.

6.4 Contrast to Prior Work

Prior Work on Adjacent Secure Communication Tools: We restate that most prior work performs in-vitro usability experiments with novices using general-purpose encryption tools (email, messaging clients). In contrast, we studied expert practitioners in-vivo focused on identity-based software signing for software supply chain security. We observe some similarities in usability factors, eg integration with existing technologies [23] and trust in the tool [23, 36, 71]. We also observe differences: our study design gives us insight into organizational and regulatory constraints, and tool adoption/migration pathways (§5.2.3, §6.1).

Generalizability to Other Software Engineering Tools: Beyond adjacent secure communication tools, our findings also speak to other classes of software engineering tools. Our findings reinforce several adoption factors noted in prior literature on open source tools [63], such as ease of use, compatibility with existing technologies, and integration flexibility, which were especially salient for identity-based signing tools like Sigstore. They also highlight 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 [59].

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 identity-based 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.

7 Threats to Validity

To discuss the limitations of our work, we follow Verdecchia *et al.*'s [33] recommendation to reflect on the greatest threats to our research. In addition, we discuss how our 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 generalizability threat is that we recruited from a population biased toward the use of Sigstore. We viewed this as necessary in order to recruit sufficient subjects. A second threat arises from the sample size of our work, which is due to the challenges of recruiting expert subjects. Third, our study relies on an expert population, whose usability concerns may not reflect those of novice users. Another 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. Sigstore has changed since then, but not the core workflows and components. We believe our findings remain relevant.

Statement of Positionality: The author team has expertise in software supply chain security in general and software signing methods in particular. They also have expertise in qualitative research methods. This background qualifies the team to design, conduct, and analyze the data described in this study. In addition, one of the authors is a Sigstore contributor. Their inclusion enabled the author team to better interpret and contextualize some of the interview data. To avoid bias, this author was not involved in developing the research design or conducting initial analysis of the results, but provided insight in later analysis.

8 Conclusion

Software signing underpins supply chain security by guaranteeing provenance and integrity, yet its effectiveness depends on usable tooling. Identity-based systems like Sigstore reduce legacy key-managed burdens of key management and signer identification, but our study shows that adoption is shaped by both strengths and frictions. Identity-based workflows, ephemeral keys, and CI/CD compatibility ease use, while integration gaps, transparency log privacy concerns, and organizational constraints remain barriers—especially in large enterprises.

By surfacing these factors, we show that usability is central to supply-chain security. Without attention to adoption barriers, even the best technical mechanisms will fail to achieve their security potential. Our findings reveal uneven maturity across Sigstore's components, and suggest lessons for other signing toolmakers, *e.g.,* OpenPubKey and SignServer, which share similar design patterns. We recommend that toolmakers prioritize official integrations and configurable privacy controls to improve usability and trust. By addressing these challenges, identity-based signing tools can expand the coverage of verifiable artifacts and attestations, shrinking supply chain attack surfaces to improve software security.

9 Acknowledgments

We thank the study participants for contributing their time to this work. We acknowledge support from Google, Cisco, and NSF #2229703. We thank Kimberly Hung for assistance with figure preparation. We also thank Hayden Blauzvern and all reviewers who provided feedback on this work.

Ethical Considerations

Our study investigates the usability of identity-based software signing tools, focusing on Sigstore as an exemplar. We conducted semi-structured interviews with 17 security practitioners across 13 organizations, analyzing their perspectives on usability strengths, weaknesses, and adoption dynamics.

Stakeholders & Potential Harms. Following the guidance of Davis *et al.* [26], we identify the following stakeholders.

- *Direct stakeholders:* (i) Interview participants, whose responses could unintentionally expose their organization's practices; (ii) The organizations employing these participants, which may face reputational, regulatory, or security risks if weaknesses are revealed; (iii) The research team, responsible for balancing transparency and confidentiality; and (iv) Adversaries, who could misuse disclosed weaknesses in signing tools or deployment practices.

- *Indirect stakeholders:* (i) Other software-producing organizations, who may be affected if the study highlights industry-wide shortcomings in the signing tool used by such organizations; (ii) Standards bodies, whose guidelines may be challenged; (iii) Software consumers, whose trust in signed artifacts could be affected; and (iv) The broader public, for whom systemic weaknesses in signing infrastructure may undermine confidence in software supply chains.

Mitigations Implemented. *Recruitment Policies:* Data were collected through voluntary interviews, with written consent obtained prior to participation. Recruitment was conducted through established professional networks; no unsolicited contact or spamming was used. Participants were free to decline or withdraw at any stage, and recruitment avoided any form of coercion. All organizations were treated equally in data collection and analysis. At no point were sensitive keys, cryptographic material, or detailed system configurations collected.

Content safeguards: All transcripts were anonymized; neither participant names nor organizational identifiers appear in published results. Findings are reported in aggregate at the thematic level (e.g., categories of usability issues) rather than specific configurations or vulnerabilities.

Privacy & data minimization: Only de-identified transcripts were retained; analysis outputs emphasize themes across participants, not individual accounts. No identifiable deployment details are included in the publication.

Operational safety: We deliberately excluded or generalized any descriptions of signing deployments that could aid adversaries, including specific weaknesses in Sigstore's components. Where risks were mentioned (e.g., transparency log privacy concerns, private instance deployment issues), results are presented at the level of usability factors rather than detailed attack vectors.

Participant Welfare. As mentioned earlier, all participants gave informed consent. Potential distress was minimal, though participants might experience discomfort when reflecting on organizational challenges. To mitigate this, questions emphasized practices and perceptions rather than failures, and responses were anonymized. No deception was involved.

Oversight and Consent Context. The academic institution's IRB reviewed and approved the protocol. Prior to consenting, all participants were informed of the study's aims and the confidentiality safeguards in place.

Decision to Proceed and Publish. We judged that the expected benefits—independent evidence on identity-based signing tool usability, insights into adoption barriers, and guidance for improving software supply chain security—outweigh residual risks under the mitigations described above. To minimize adversarial risk, we publish only aggregated themes, not sensitive technical details, to ensure that our findings strengthen rather than weaken trust in signing infrastructures.

Open Science

We acknowledge that USENIX Security has an open science policy: that authors are expected to openly share their research artifacts by default. The research artifacts associated with this study are:

- Raw transcripts of interviews
- Anonymized transcripts of interviews
- Interview protocol
- Codebook

Things we have shared: The *interview protocol* is a crucial part of any interview study, since it allows for the critical review of a study design as well as its replication . Since it is semi-structured, we include all of the questions asked of all subjects, as well as examples of the follow-up questions we asked. We also share the *codebook*, with codes, definitions, and example quotes that we coded for each code.

Things we cannot share: For subject privacy reasons, and for IRB compliance, we cannot share the raw transcripts. We also choose not to share the anonymized transcripts of the interviews. Given the high organizational ranks of many of our subjects, and the small size of the subject pool resulting from our recruiting strategy, we believe there is a high risk of de-anonymization even of anonymized transcripts. Therefore, we do not share the anonymized transcripts.

Artifact Repository: An artifact containing our interview protocol, codebook, etc. is available at: https://doi.org/10.5281/zenodo.17969423

References

[1] The GNU privacy guard. https://www.gnupg.org/, December 2024.

[2] 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 (SCORED)*, 2022.

[3] A. R. Lyon et al. The cognitive walkthrough for implementation strategies (cwis): a pragmatic method for assessing implementation strategy usability. *Implementation Science Communications*, 2(1):78, 2021.

[4] Amazon Web Services. AWS signer developer guide. https://docs.aws.amazon.com/signer/latest/developerguide/Welcome.html, 2024. Accessed: 2025-05-30.

[5] S. Baltes and P. Ralph. Sampling in software engineering research: A critical review and guidelines. *Empirical Software Engineering (EMSE)*, 2022.

[6] S. Benthall. Assessing software supply chain risk using public data. In *IEEE Software Technology Conference (STC)*, 2017.

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

[8] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 2006.

[9] C. Braz and J. Robert. Security and usability: The case of the user authentication methods. In *Conference on l'Interaction Homme-Machine*. ACM, April 2006.

[10] 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 (SCORED)*, 2022.

[11] C. Rusu et al. User experience evaluations: Challenges for newcomers. In Aaron Marcus, editor, *Design, User Experience, and Usability: Design Discourse*. Springer, Cham, 2015.

[12] R. J. Chenail. Interviewing the investigator: Strategies for addressing instrumentation and researcher bias concerns in qualitative research. *Qualitative report*, 16(1):255–262, 2011.

[13] Cloud Native Computing Foundation. Software supply chain best practices, May 2021. https://github.com/cncf/tag-security/blob/93450ff2a67d63562d56fdc9f336965bf1615863/community/working-groups/supply-chain-security/supply-chain-security-paper/CNCF_SSCP_v1.pdf.

[14] D. Cooper et al. Security considerations for code signing. *NIST Cybersecurity White Paper*, January 2018.

[15] E. Heilman et al. OpenPubkey: Augmenting OpenID connect with user held signing keys, 2023. https://eprint.iacr.org/2023/296.

[16] A. Al Hadwer et al. A systematic review of organizational factors impacting cloud-based technology adoption using technology-organization-environment framework. *Internet of Things*, 15:100407, 2021.

[17] A. Reuter et al. Secure email - a usability study. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 36–46, Cham, 2020. Springer International Publishing.

[18] B. Xia et al. An empirical study on software bill of materials: Where we stand and the road ahead. In *IEEE/ACM International Conference on Software Engineering (ICSE)*, May 2023.

[19] C. Carroll et al. "Best fit" framework synthesis: Refining the method. *BMC Medical Research Methodology*, 13(1):37, 2013.

[20] C. L. Okafor et al. Diverify: Diversifying identity verification in next-generation software signing. *arXiv preprint arXiv:2406.15596*, 2024.

[21] D. Cooper et al. Protecting software integrity through code signing. Technical Report ITL Bulletin, National Institute of Standards and Technology, May 2018.

[22] D. I. K. Sjoberg et al. The future of empirical methods in software engineering research. In *2007 Future of Software Engineering (FOSE)*. IEEE, 2007.

[23] E. Atwater et al. Leading Johnny to water: Designing for usability and trust. In *Symposium on Usable Privacy and Security (SOUPS)*, July 2015.

[24] G. Terry et al. Thematic analysis. *The SAGE handbook of qualitative research in psychology*, 2, 2017.

[25] H. S. Bansal et al. "Migrating" to new service providers: Toward a unifying framework of consumers' switching behaviors. *Journal of the Academy of Marketing Science*, 33(1), 2005.

[26] J. C. Davis et al. A guide to stakeholder analysis for cybersecurity researchers. *arXiv preprint arXiv:2508.14796*, 2025.

[27] J. Kjeldskov et al. Does time heal? A longitudinal study of usability. In *Australian Computer-Human Interaction Conference 2005 (OzCHI)*, 2005.

[28] K. Merrill et al. Speranza: Usable, privacy-friendly software signing. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023.

[29] M. Lennartsson et al. Exploring the meaning of usable security–a literature review. *Information & Computer Security*, 29(4):647–663, 2021.

[30] N. K. Gale et al. Using the framework method for the analysis of qualitative data in multi-disciplinary health research. *BMC Medical Research Methodology*, 13:117, 2013.

[31] P. Ladisa et al. SoK: Taxonomy of attacks on open-source software supply chains. In *IEEE Symposium on Security and Privacy (S&P)*, 2023.

[32] R. Chandramouli et al. *Strategies for the integration of software supply chain security in DevSecOps CI/CD pipelines*. US Department of Commerce, National Institute of Standards and Technology, 2024.

[33] R. Verdecchia et al. Threats to validity in software engineering research: A critical reflection. *Information and Software Technology*, 164, 2023.

[34] S. Crowe et al. The case study approach. *BMC Medical Research Methodology*, 11:100, June 2011.

[35] S. Easterbrook et al. *Selecting Empirical Methods for Software Engineering Research*, page 285–311. Springer, London, 2008.

[36] S. Fahl et al. Helping Johnny 2.0 to encrypt his Facebook conversations. In *The Eighth Symposium on Usable Privacy and Security (SOUPS)*, July 2012.

[37] S. Ruoti et al. Why Johnny still, still can't encrypt: Evaluating the usability of a modern PGP client. *arXiv preprint arXiv:1510.08555*, 2015.

[38] S. Ruoti et al. Private Webmail 2.0: Simple and easy-to-use secure email. In *Symposium on User Interface Software and Technology (UIST)*, 2016.

[39] S. Torres-Arias et al. in-toto: Providing farm-to-table guarantees for bits and bytes. In *28th USENIX Security Symposium (USENIX Security 19)*, August 2019.

[40] T. Lorey et al. Social science theories in software engineering research. In *International Conference on Software Engineering*, ICSE '22, July 2022.

[41] G. C. Feng. Intercoder reliability indices: disuse, misuse, and abuse. *Quality & Quantity*, 2014.

[42] 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.

[43] D. M. Fernández and J. Passoth. Empirical software engineering: From discipline to interdiscipline. *Journal of Systems and Software*, 148:170–179, 2019.

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

[45] J. Gerring. What is a case study and what is it good for? *American Political Science Review*, 98(2):341–354, May 2004.

[46] Matthew Green and Matthew Smith. Developers are not the enemy!: The need for usable security apis. *IEEE Security & Privacy*, 14(5), 2016.

[47] A. Hicks. SoK: Log based transparency enhancing technologies, 2023. https://arxiv.org/abs/2305.01378.

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

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

[50] 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 Symposium*, 2024.

[51] in toto and The Linux Foundation. in-toto: A framework to secure the integrity of software supply chains. https://in-toto.io/, 2023.

[52] 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, 1997.

[53] J. L. Campbell et al. Coding in-depth semistructured interviews: Problems of unitization and intercoder reliability and agreement. *Sociological methods & research*, 42(3):294–320, 2013.

[54] K. Cresswell et al. Developing and applying a formative evaluation framework for health information technology implementations: Qualitative investigation. *J Med Internet Res*, 22(6), 2020.

[55] K. G. Kalu et al. An industry interview study of software signing for supply chain security. In *USENIX Security Symposium*, 2025.

[56] K. MacDorman et al. An improved usability measure based on novice and expert performance. *International Journal of Human–Computer Interaction*, 27(3), 2011.

[57] 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*, 24(1):182, 2024.

[58] S. M. Larson. Python and Sigstore. https://sethmlarson.dev/python-and-sigstore, 2024. Accessed: 2025-05-30.

[59] Valentina Lenarduzzi et al. Open source software evaluation, selection, and adoption: a systematic literature review. In *Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2020.

[60] J. R. Lewis. Usability: Lessons learned...and yet to be learned. *International Journal of Human-Computer Interaction*, 30(9), 2014.

[61] M. A. Sasse and I. Flechais. Usable security: Why do we need it? How do we get it? O'Reilly, 2005.

[62] J. C. Mankins. Technology readiness assessments: A retrospective. *Acta Astronautica*, 65(9-10), 2009.

[63] Del Nagy et al. Organizational adoption of open source software: barriers and remedies. *Commun. ACM*, 53(3):148–151, March 2010.

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

[65] OpenPubKey. OpenPubKey: Issues. https://github.com/openpubkey/openpubkey/issues, 2025. GitHub issues page.

[66] C. O'Connor and H. Joffe. Intercoder reliability in qualitative research: Debates and practical guidelines. *International Journal of Qualitative Methods*, 2020.

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

[68] Red Hat. What is Skopeo? https://www.redhat.com/en/topics/containers/what-is-skopeo, July 2022. Published July 18, 2022.

[69] B. H. Rowlands. Grounded in practice: Using interpretive research to build theory. *Electronic Journal of Business Research Methods*, 3(1):pp81–92, 2005.

[70] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering (EMSE)*, 14(2), 2009.

[71] S. Ruoti et al. Confused Johnny: When automatic encryption leads to confusion and mistakes. In *Symposium on Usable Privacy and Security (SOUPS)*, July 2013.

[72] S. Sheng et al. Why Johnny still can't encrypt: Evaluating the usability of email encryption software. In *Symposium on Usable Privacy and Security (SOUPS)*. ACM, 2006.

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

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

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

[76] Sigstore. Securing your software supply chain without changing your devops workflow. https://blog.sigstore.dev/securing-your-software-supply-chain-without-changing-your-devops-workflow-e23393a5fffa, December 2022.

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

[78] Sigstore. Using Sigstore to meet FedRAMP compliance at Autodesk. https://blog.sigstore.dev/using-sigstore-to-meet-fedramp-compliance-at-autodesk-6f645a920abc, November 2022.

[79] Sigstore Blog. Announcing NPM support: Public beta. https://blog.sigstore.dev/npm-public-beta/, 2024. Accessed: 2025-05-30.

[80] Sonatype. Working with PGP signatures. https://central.sonatype.org/publish/requirements/gpg/. Accessed: 2025-05-30.

[81] 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.

[82] R. E. Stake. *The Art of Case Study Research*. SAGE Publications, Thousand Oaks, CA, 1995.

[83] C. G. Steeh. Trends in nonresponse rates, 1952–1979. *Public Opinion Quarterly*, 45(1), Spring 1981.

[84] 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.

[85] T. Bruckhaus et al. The impact of tools on software productivity. *IEEE Software*, September 1996.

[86] 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*, 30(2), 2022.

[87] T. Kuppusamy et al. PEP 480 – Surviving a compromise of PyPI: End-to-end signing of packages, oct 8 2014. https://peps.python.org/pep-0480/.

[88] T. R. Schorlemmer et al. Signing in four public software package registries: Quantity, quality, and influencing factors. In *IEEE Symposium on Security and Privacy (S&P)*, May 2024.

[89] T. R. Schorlemmer et al. Establishing provenance before coding: Traditional and next-generation software signing. *IEEE Security & Privacy, IEEE S&P Magazine '25*, (01), 2025.

[90] T. S. Andre et al. Testing a framework for reliable classification of usability problems. In *The Human Factors and Ergonomics Society Annual Meeting*, volume 44. SAGE Publications, 2000.

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

[92] 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/.

[93] Mary Theofanos and Whitney Quesenbery. Towards the design of effective formative test reports. *J. Usability Studies*, 1(1):27–45, November 2005.

[94] 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*, 24(2):138–161, 2023.

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

[96] A. Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *USENIX Security Symposium*, volume 348, 1999.

[97] M. Willett. Lessons of the SolarWinds hack. In *Survival April–May 2021: Facing Russia*, pages 7–25. Routledge, 2023.

[98] Yahoo Inc. Scaling up supply chain security: Implementing Sigstore for seamless container image signing. https://www.yahooinc.com/paranoids/scaling-up-supply-chain-security-implementing-sigstore-for-seamless-container-image-signing, 2023. Accessed: 2025-05-30.

[99] R. K. Yin. *Case study research and applications: Design and methods*. SAGE, sixth edition, 2018.

[100] Z. Newman et al. Sigstore: Software signing for everybody. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[101] V. Zimmer and M. Krau. Establishing the root of trust. https://uefi.org/sites/default/files/resources/UEFI%20RoT%20white%20paper_Final%208%208%2016%20(003).pdf, 2016. Accessed 26 May, 2025.

A Code Saturation

We present a saturation analysis of our studies in Figure 6. The orange trendline (saturation curve) was measured by identifying the cumulative unique codes present in each interview. We observe saturation at the 12th interview, indicating that no new unique codes were identified after participant 12. Additionally, the green trendline (number of codes) shows that the lowest number of codes was obtained from subject P4. This is also reflected in the blue trendline (unique number of codes).

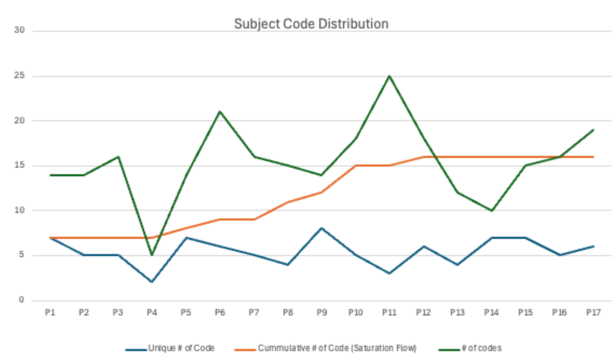


Figure 6: Saturation curve. Interviews are plotted in the order in which they were conducted. Each interview covered many topics in detail (orange line, blue line). However, as the green line shows, our results saturated (*i.e.,* stopped observing new perspectives) around interview 12.

B Organizational Demographics

We summarize additional organizational demographic information of our subjects in Table 6. This table provides further context for the subject's organizational size, type of software product, and how many of our subjects belonged to each organization.

Table 6: Organizational Demographics. For anonymity, we only highlight the number of subjects in each organizational category. Letters A-M are used to denote each organization.

Type	Breakdown (#Organizations|#Subjects)
Organizational Size (Employee Size)	Small (<100) (4/6), Medium (<1500) (3/4), Large (>1500) (6/7)
Product Area	Digital technology (1/3), SSC Security (2/4), Social technology (1/1), Dev tools (1/1), Telecommunications (1/1), Cloud security (2/3), Aerospace Security (1/1), Internet services (2/2), Cloud + OSS Security (1/1), Cloud + Dev tools (1/1)
Subject Distribution	A (2), B (3), C (1), D (1), E (2), F (1), G (1), H (1), I (1), J (1), K (1), L (1), M(1)

C Technical Report

A extended version of our paper is available at https://arxiv.org/abs/2503.00271. The primary difference is its appendices, which contain the following information:

- Appendix A: Traditional software signing workflow (diagram and narrative description).
- Appendix B: The full interview protocol used in our study.
- Appendix C: The codebooks used in our analysis, with illustrative quotes mapped to each code.
- Appendix D: Code saturation analysis.
- Appendix E: Survey results that were omitted from the main paper due to low-quality data.
- Appendix E.1: Sigstore components mostly used by our sample population.
- Appendix E.2: Other results omitted from the paper due to space constraints.
- Appendix E.3: Expanded demographic table.
- Appendix E.4: Summary of factors influencing Sigstore adoption for Sigstore users (prior to adoption).
- Appendix E.5: Summary of factors influencing consideration of Sigstore among non-Sigstore users.
- Appendix E.6: Additional discussions and implications of results.
- Table 13: Driver labels (Push/Pull/Barrier) mapped to concrete codes with factor tags (T/P/O/M) and example participant IDs.