



# The Limits of Education in Driving Adoption of Security Technologies

**Carolina Carreira**

PhD Thesis Proposal

## Thesis Proposal Committee

| | |
|---|---|
| Chairperson: | Prof. Nuno Nunes |
| Supervisors: | Prof. João F. Ferreira<br>Prof. Alexandra Mendes<br>Prof. Nicolas Christin |
| Committee members: | Prof. Lorrie Cranor<br>Prof. Michelle Mazurek<br>Prof. Nuno Nunes |

**April 2026**

**Abstract**

Trusted Execution Environments, formally verified software, and verification-aware programming languages offer technical guarantees such as hardware-level isolation, functional correctness, and verifiable program behavior. However, user adoption remains low and these technologies are hard to use by developers. In this thesis, through user studies of both developers and end users, I will show that user and developer education does improve general understanding of security tools and formal verification, but presents fundamental limitations that make it unsuitable as the only tactic for increasing the adoption of reliable security technologies. For instance, some communication strategies improve comprehension of security technologies but may still fail to increase adoption when users' core concerns (human misuse, data selling, long-term retention) fall outside the technology's scope (see Section 2.1.2). Moreover, intuition framing can backfire because shallow explanations that match users' mental models (e.g., "mathematically correct") may increase adoption but risk over-promising guarantees (see Section 2.1.3). Regarding developer challenges, our results suggest that learning curves are addressable through education, but interventions should promote active learning rather than passive reliance on tools (see Sections 2.2.1 and 2.2.2). Together, these findings suggest that closing the adoption gap requires communication strategies that engage with the concerns users hold.

**Contents**

## 1 Introduction

Security technologies have made progress in recent years and are increasingly more reliable. For example, Trusted Execution Environments (TEEs) offer hardware-level isolation, enabling computation on sensitive data even in untrusted cloud environments [48]. Confidential computing and TEEs have applications in AI and machine learning [16,40,53], IoT [5], and blockchain smart contracts [54]. Advances in formal verification, an approach that can mathematically prove the correctness of software, providing guarantees that go beyond what testing can achieve, have also contributed to the increased reliability of security technologies. Examples include verified compilers like CompCert [38], verified implementations of cryptographic protocols used in production systems such as Amazon's s2n [18], and verified password managers [10, 25].

However, despite these improvements and advances, the adoption of reliable security technologies and of formal verification remains low. On the one hand, technologies such as TEEs are available in commodity hardware from major manufacturers, but most users and many developers have never heard of them, let alone understand what they guarantee [41]. On the other hand, formal verification is rarely used outside of safety-critical niches [17, 32]. Reliable security technologies have gained visibility through high-profile deployments (e.g., Apple's use of differential privacy) [4], but end users struggle to understand what protections they actually provide [19,50], and developers face significant barriers to integrating them correctly [22]. There is a gap between what these technologies can offer and how widely they are actually used.

A possible way to address this problem is education. The premise is that if users do not adopt secure technologies because they do not understand them, then better explanations should close the gap. If developers do not build with verification-aware tools because the learning curve is too steep, then better pedagogy and tooling should help. This logic has motivated a substantial body of work in usable security, from the design of security warnings [9] and privacy nutrition labels [33] to educational interventions aimed at developers [17].

Through a series of user studies involving both end users and developers, I investigate the relationship between education, comprehension, and adoption across three areas: end-user understanding of Trusted Execution Environments, user perception of formally verified software, and developer practice with verification-aware programming languages. I propose the following thesis statement:

> Current user and developer education does improve general understanding of security technologies and formal verification, but presents fundamental limitations that make it unsuitable as the only tactic for increasing the adoption of reliable security technologies.

More specifically, this thesis demonstrates three findings:

**1. User explanations face a limitation.** Even when explanation strategies effectively improve comprehension of security technologies, they may still fail to increase willingness to adopt when users' core concerns (e.g., human misuse of data, long-term retention, and third-party data selling) fall outside the scope of what the technology protects [39]. This limitation stems from a mismatch between the technology and the threats users consider important.

**2. Intuition framing can backfire.** Shallow explanations that align with users' intuition, such as framing formal verification as "mathematically correct", may increase willingness to adopt, but they do so by exploiting prior associations that may not correspond to the technology's actual guarantees [11]. Users who accept such framings may over-attribute protection to general security guarantee that the formal verification does not provide.

**3. Addressing steep developer learning curves requires active engagement.** The principal barriers that developers face with verification-aware languages (i.e., difficulty formulating valid proof strategies, inadequate error feedback, and insufficient educational resources) [43] are obstacles that targeted educational interventions can address. However, for interventions to be effective they should promote active engagement with the verification task. Passive reliance on automated aids, including large language model assistance, may fail to develop the proof intuition that constitutes the long-term adoption barrier [13].

**1.1 Overview of Completed Studies** To support this thesis, I present five completed studies and one proposed study, organized around the two sides of the adoption problem: end-user understanding and developer practice.

**Systematic Review on Security Communication [12].** To ground this thesis in existing knowledge, I conduct a systematic literature review of how security properties are communicated to end users. The review maps communication strategies across 114 relevant studies, identifies the outcomes that have been measured, comprehension, trust, and behavior change, and distills design considerations and open challenges for security communication (Section 2.1).

**Communicating TEE Guarantees to End Users [39].** This study designs and evaluates explanation strategies for TEEs, with varying technical detail and framing. Non-technical, threat focused explanations significantly improve comprehension of TEE capabilities, however, willingness to adopt TEE-enhanced technologies and perceptions of data safety remain largely unchanged across explanation conditions. Qualitative analysis reveals that participants' primary concerns (human misuse of data, data selling, and long-term retention) fall outside the scope of what TEEs protect against, establishing that improved comprehension alone does not drive adoption when users' core concerns are not addressed by the technology (Section 2.1.2).

**Formal Verification and Users' Willingness to Adopt Password Managers [11].** This study examines whether explaining that a password manager has been formally verified increases users' willingness to adopt it. Users generally have no prior understanding of formal verification; a brief, intuition-aligned explanation framing it as "mathematically correct" significantly increases willingness to adopt. However, participants who receive this explanation consistently over-attribute the scope of the guarantee, interpreting mathematical correctness as a general security assurance that formal verification of specific components does not provide. These results demonstrate that security communication can increase adoption while simultaneously producing inflated expectations that exceed the technology's actual guarantees (Section 2.1.3).

**Challenges Developers Face with Verification-Aware Languages [43].** On the developer side, this study identifies the challenges that practitioners encounter when using verification-aware programming languages such as Dafny, combining a structured literature review, analysis of Stack Overflow discussions, and an online survey of 31 practitioners. The study documents that the most prevalent barriers cluster around proof strategy formulation and inadequate tool feedback rather than syntax or environment configuration, and that these obstacles persist across experience levels (Section 2.2.1).

**LLMs as Educational Aids for Formal Verification [13].** Building on the barriers identified in the previous study, this study investigates whether large language models can serve as effective educational aids for students learning deductive program verification with Dafny. A controlled study with 14 master's students demonstrates that LLM assistance substantially improves performance, however, the benefits are strongly modulated by engagement strategy. Students who provide context rich prompts and exercise independent reasoning over the model's output achieve the highest outcomes, while those who defer to the model obtain weaker results and do not show the proof intuition that formal verification requires (Section 2.2.2).

## 1.2 Future Work

To further substantiate the thesis, I propose one additional study.

**Explanations for Formally Verified Software.** The completed end-user studies isolate two complementary failure modes of security technology communication. The TEE study demonstrates that well-designed explanations can improve comprehension without affecting willingness to adopt when users' concerns lie outside the technology's protection boundary. The formal verification willingness study demonstrates the converse failure: shallow, intuition-aligned framing increases willingness to adopt while inducing systematic over-attribution of the guarantee's scope. This study tests whether explanation depth can decouple these two outcomes by varying the depth and precision of formal verification explanations across four compositional conditions, ranging from no explanation to a detailed, scope-delineating description of what formal verification establishes and what it does not. The study employs four consumer facing scenarios spanning password management, secure messaging, AI-assisted fi-

nance, and health data sharing, recruits through Prolific, and evaluates comprehension, willingness to adopt, and over-attribution. It provides cross-domain evidence on the scope limitation and determines whether accurate scope delineation is compatible with the adoption gains that shallow framing produces (Section 3.1).

**1.3 Thesis Scope** Overall, this thesis sits at the intersection of usable security and formal methods. On the user side, the focus is on explanation design: how security properties can be communicated to non-expert users, and whether improved communication leads to behavior change. On the developer side, the focus is on **educational interventions** and how developers can be supported in learning to use verification tools, and what role AI-assisted tools can play.

## 2 Completed Work

The completed work is organized around the two dimensions of the adoption problem identified in Section 1. Sections 2.1, 2.1.2 and 2.1.3 address end-user understanding of security guarantees and Sections 2.2.1 and 2.2.2 addresses developer practices with verification-aware languages.

**2.1 End-User Communication** The following studies examine end-user communication, first by synthesizing prior research on security communication, then by evaluating how TEEs can be explained to non-experts, and finally by analysing how formal verification influences willingness to adopt password managers.

### *2.1.1 Systematic Review on Security Communication*

This section presents the systematic review on security communication strategies [12], which aims to map how security properties are communicated to end users. Through a structured and reproducible literature review of over 3,400 candidate papers, we identified 114 relevant studies and synthesized their findings into design considerations and open challenges for security communication. This review provides the conceptual foundation for the empirical studies that follow in this thesis.

Security communication encompasses the methods used to convey security information to users, including warnings and notifications, privacy policies, explanations, security advice, and educational materials. Examples range from browser SSL certificate warnings and phishing alerts to mobile app permission dialogs. The effectiveness of these communications directly affects whether users can recognize threats, make informed decisions, and adopt protective behaviors [2, 45].

Despite the widespread availability of security information, users frequently struggle to act on it due to incomplete information, conflicting advice, and the inherent tension between security and usability [2]. Prior work has investigated various aspects of security communication across domains, security warnings [9], privacy policies [14], authentication interfaces [23], and mobile permissions [51], but this research remains fragmented. Previous surveys have

partially addressed the space: Hancock et al. conducted a meta-analysis limited to security warnings [28], Lennartsson et al. reviewed usable security more broadly [37], and Chaudhary et al. focused specifically on password managers [15]. These efforts are domain-specific, which results in a fragmented picture that does not address cross-domain research questions.

We conducted a structured and reproducible systematic literature review following established guidelines. We searched four primary databases for computer science literature: Scopus, ScienceDirect, Web of Science, and ACM Digital Library. Our search was based on keywords spanning three dimensions: communication (e.g., understanding, explanation, describing, advice), usability (e.g., user study, usability), and security (e.g., secure) applied to the Title, Abstract, and Author Keywords fields. The search was executed systematically across all four databases on September 29th, 2025. In total, 3,485 papers were obtained across the four databases. After removing 2,734 duplicates using Rayyan, a collaborative systematic review tool, we applied inclusion and exclusion criteria in two phases. Inclusion criteria required peer-reviewed articles, conference papers, and book chapters focused on security communication to users, published in English, and available online. We excluded studies focusing on technical security aspects (such as cryptography or intrusion detection) and those unrelated to security communication. In the first phase, two independent reviewers double-anonymized and screened the titles and abstracts, with a third reviewer resolving disagreements. After this phase, 134 papers remained. In the second phase, one of the authors reviewed the full texts of all remaining articles and excluded 20 based on the exclusion criteria. We finalized our selection of 114 papers, which form the corpus for this review. We addressed four research questions: (RQ1) what research techniques are used to study users of security communication, (RQ2) which communication techniques are used to communicate about security, (RQ3) what security communication problems are identified, and (RQ4) what design considerations does the literature provide for designing effective security communication.

Our analysis goes through the following research questions.

**Research Techniques (RQ1).** The field is dominated by user-centered research designs, predominantly surveys and interviews. Qualitative studies commonly used double-anonymized coding procedures with open, emergent coding. Quantitative studies relied on standard statistical tests such as ANOVA, Kruskal-Wallis, and Mann-Whitney U tests. The majority of our corpus focuses on the short term with single-time studies and does not examine security communication longitudinally. Short-term studies may miss changes in behavior over time due to new threats or increased user education, and they may capture immediate reactions but fail to assess how well users retain and apply security knowledge over time. The corpus also revealed a significant reproducibility gap: only 28% of user studies shared their study protocols, limiting the ability to replicate and build upon prior work.

**Communication Modality (RQ2).** The most frequent type of security communication studied was descriptions and explanations, appearing in 47% of papers (n = 54). Studies in this category evaluated the effectiveness of various methods for conveying security information

to enhance user comprehension and behavior. Key themes include browser interface explanations (where research consistently reveals user misconceptions about SSL certificates and private browsing modes), security terminology, and mobile permission explanations (where users display persistently low attention and comprehension rates). The second most popular category was warnings and notifications (21%, n = 25).

**Communication Problems (RQ3)** . We identified specific user difficulties, including information overload, comprehension of technical jargon, and balancing security awareness with user comfort. From these, we conceptualized two recurring trade-offs. The first is the Comprehension Jargon trade-off: users seem to want to understand and learn more about technology, and some studies suggest that technical language can enhance credibility. However, overly technical jargon can hinder understanding and trust. The second is the Awareness–Discomfort trade-off: users want to understand the risks and benefits of security technology, which can help them make more informed choices, but explaining threats in too much detail can make users feel unsafe or anxious. Explaining too little is also not useful, as it does not effectively inform users. These trade-offs are particularly relevant to this thesis, as they directly shaped our explanation design in the TEE study (Section 2.1.2) and the formal verification willingness study (Section 2.1.3).

**Design Considerations (RQ4).** We distilled five design considerations from the corpus. Collectively, they indicate that effective security communication depends on salience, concise explanation, adaptation to user characteristics, adaptation to task context, and empirical validation.

**C1: Design for attention** Security communication should attract attention through clear visual structure and appropriate salience, while avoiding unnecessary interruption and habituation.

**C2: Embed education in use** Security communication should explain relevant concepts and rationales in a brief, contextual form that supports understanding without imposing substantial additional effort.

**C3: Adapt to user characteristics** Security communication should be calibrated to user expertise, security sensitivity, and cultural context, since a single formulation does not serve all audiences equally well.

**C4: Adapt to task context** Security communication should reflect the decision context, including the user's immediate goal, timing, and environment, so that information is presented when it can inform action.

**C5: Test with representative users** Security communication should be evaluated with representative users before deployment to determine whether it is noticed, understood, and effective.

Our results also revealed a gap between principle and practice: of the 114 studies, the majority describe new communication methods (66%, n = 75) rather than evaluating existing deployed systems, suggesting that even widely accepted principles face implementation barriers.

Our systematic review summarizes part of the security communication research. However, we found that the technologies at the center of this thesis, TEEs and formal verification, are absent from the existing literature on security communication. This gap directly motivates the empirical studies in Sections 2.1.2 and 2.1.3.

The review also provides methodological grounding for the subsequent studies. The design considerations and trade-offs identified here, particularly the Comprehension–Jargon and Awareness–Discomfort trade-offs, informed our approach to designing TEE explanations (Section 2.1.2) and our proposed future work.

Moreover, the review revealed that the existing literature predominantly treated comprehension as the primary outcome of interest. Studies tended to measure whether users understood or noticed a given communication. What is less studied is whether improved comprehension leads to changes in behavior or willingness to adopt. There is a gap between comprehension as a measured outcome and adoption as the desired outcome.

> We finished a version of this review and shared it on arXiv, but since then, we have expanded it with newer papers and are finishing the recoding. We plan to submit this work to NDSS 2027.

### *2.1.2 Communicating TEE Guarantees to End Users*

Trusted Execution Environments (TEEs) represent a class of hardware-based security technologies that provide isolation guarantees for sensitive computations. TEEs ensure that data and code remain confidential and integral against a broad threat model that includes compromised operating systems, firmware, and other software components on the same hardware [48]. While TEE mechanisms exist in commodity processors from major manufacturers (e.g., ARM TrustZone, Intel SGX, and AMD SEV) user awareness remains limited [41]. Prior work has suggested that in domains such as cloud storage and smart home IoT, informing users about the presence of TEEs may increase their comfort with data collection [41]. However, that work did not investigate *how* to communicate TEE capabilities to non-experts, leaving a methodological gap: while the existence of TEEs appears to matter for user comfort, guidance on explaining their guarantees and limitations to achieve that effect is absent from the literature.

We investigated which explanation strategies effectively communicated TEE guarantees to end users. The central research question was whether improving comprehension of what TEEs protect translated into increased willingness to use TEE-enhanced services and higher perceptions of data safety.

We conducted two sequential online surveys with 966 unique participants recruited through Prolific, a crowdsourcing platform. The surveys employed a between-subjects design to evaluate TEE explanations while holding other factors constant. Participants were presented with

realistic scenarios in two domains identified as TEE use cases: medical research and smart home IoT. Each scenario described data collection for either a simple analysis task or a complex AI task, creating four scenario families. For each scenario, participants read a randomly assigned explanation of how TEEs protect that data.

The explanation strategies tested in Survey 1 were constructed by independently varying three dimensions:

1. **Focus (sentence 1):** How the TEE is framed: as a hardware component (*Hardware*), as a trust mechanism (*Trust*), or with minimal justification (*Unsubstantial*).
2. **Technicality (sentence 2):** The level of technical language: non-technical plain language (*Non-Technical*) or terminology-heavy descriptions (*Technical*).
3. **Threat framing (sentence 3):** Whether the explanation emphasizes specific attacks the TEE *Prevents* or makes no mention of prevention (*No Prevents*).

These dimensions lead to twelve candidate explanations. This combinatorial approach operationalizes the Comprehension–Jargon and Awareness–Discomfort trade-offs identified in the systematic review. Survey 2 refined the design based on initial results: we retained the most effective explanation variants (Non-Technical framing combined with threat-focused content) and introduced supplementary material in the form of a frequently asked questions (FAQ) document. The FAQ addressed the most common questions from Survey 1 participants, specifically about TEE mechanisms, verification, and real-world deployment, positioning it as optional technical detail.

**Explanation Comprehension** Across both surveys, comprehension was assessed using 10 true/false questions that measured understanding of TEE capabilities and limitations. Survey 1 revealed an asymmetry: participants correctly answered 86.5% of questions about TEE *features* (properties TEEs actively protect, such as confidentiality of data in the TEE) but only 71.5% of questions about *limitations* (scenarios TEEs do not protect against, such as legitimate users misusing accessible data). This pattern held across both scenarios and both survey samples (see Table 1).

Explanation design significantly influenced comprehension for feature questions. Non-technical explanations were particularly effective for questions about data access control (increasing correct responses for the relevant question by a statistically significant margin across medical scenarios). Threat-focused explanations that explicitly stated which attacks TEEs *prevent* substantially improved comprehension of malware protection, with effects significant across both medical scenarios and the complex smart home scenario. In contrast, technical jargon and abstract framings (such as those emphasizing general "trust" or "isolation" without concrete examples) did not improve comprehension and sometimes hurt it.

Survey 2 results were mixed. The FAQ improved comprehension for feature-related questions and FAQ-specific questions (77% of participants, even in the "hidden by default" condition, expanded the FAQ at least once), however, providing the FAQ sometimes decreased correct responses to limitation questions.

| | *Features* | | | | | *Limitations* | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Variable | **Q1** | **Q2** | **Q3** | **Q4** | **Q5** | **Q6** | **Q7** | **Q8** | **Q9** | **Q10** |
| McFadden's $R^2$ | 0.063 | 0.104 | 0.029 | 0.072 | 0.024 | 0.022 | 0.020 | 0.012 | 0.032 | 0.030 |
| *Sentence 1 [Baseline = Unsubstantial]* | | | | | | | | | | |
| Hardware | 1.52 | 0.38 | 1.32 | 0.99 | 0.43 | 1.03 | 0.47 | 0.91 | 0.41** [0.210, 0.770] | 1.28 |
| Trust | 1.46 | 0.54 | 1.97 | 0.99 | 0.56 | 1.51 | 0.70 | 1.51 | 0.79 | 0.93 |
| *Sentence 2 [Baseline = Technical]* | | | | | | | | | | |
| Non-technical | 1.17 | 8.50** [2.18, 56.66] | 0.93 | 1.19 | 1.39 | 1.63 | 0.89 | 1.15 | 1.08 | 1.22 |
| *Sentence 3 [Baseline = No Prevents]* | | | | | | | | | | |
| Prevents | 0.68 | 1.11 | 1.89 | 3.67*** [1.78, 8.14] | 1.04 | 0.90 | 0.74 | 0.99 | 1.14 | 0.53 |

***$p < 0.001$; **$p < 0.01$; *$p < 0.05$. Odds ratios from logistic regression; 95% CI shown for significant predictors.

**Table 1:** Logistic regression odds ratios for each comprehension question in the TEE study (Medical Without AI scenario). Columns Q1–Q5 concern TEE features; Q6–Q10 concern TEE limitations. Non-technical framing substantially improved comprehension of data access control (Q2, OR = 8.50, $p < 0.01$); threat-focused framing improved comprehension of malware prevention (Q4, OR = 3.67, $p < 0.001$). Hardware framing reduced correct responses for a limitation question (Q9, OR = 0.41, $p < 0.01$). McFadden's $R^2$ is reported per question.

Overall, the comprehension results validate that clear, non-technical, threat-focused explanations do improve understanding of what TEEs protect.

**Willingness and Safety Perceptions** Despite improvements in comprehension, willingness and safety perceptions remained stable across conditions. In Survey 1, across all explanation variants and scenarios, approximately 20–22% of participants reported being "definitely willing" to participate in or adopt the TEE-protected service, and 50–56% were "maybe willing", these distributions were statistically indistinguishable across explanation conditions. Similarly, 24–28% of participants believed their data would be "completely safe" and 62–67% felt it would be "somewhat safe," again with no significant differences between most explanation variants. The only exception was a small effect of non-technical framing on safety perceptions in the simple smart home scenario, but it was modest and did not translate into increased willingness.

Survey 2 showed that providing an FAQ had minimal effects on willingness or safety perceptions. Participants who received the FAQ showed no statistically significant increase in willingness to adopt services (with one minor exception in a simple smart home scenario), which suggests that supplementary technical information may not bridge the comprehension-to-adoption gap.

This finding contradicts the assumption that education leads to more adoption. Participants in this study achieved high comprehension of TEE features through well-designed non-technical explanations, yet their reported willingness to use TEE-protected services remained unchanged.

**Other User Concerns** To understand why willingness did not improve despite successful comprehension, we conducted qualitative analysis of the 310 open-ended questions asked by 252 participants (66% of the survey population). The participants' responses suggest that primary data privacy concerns fell *outside the scope of what TEEs protect*.

The most frequent concern category involved data practices by humans with legitimate system access (41 participants asked about data retention policies; 23 asked whether researchers or developers would misuse data; 27 raised concerns about external attackers like hackers). Another category addressed general distrust of technology ecosystems: 16 participants questioned whether people trust technology at all; 10 expressed skepticism that any system could truly keep data safe. Notably, 10 participants explicitly stated discomfort with the term "trusted" in "Trusted Execution Environment," interpreting it as a marketing claim.

These concerns are not addressable by TEE technology. TEEs prevent unauthorized access by malicious software and provide data confidentiality against hardware-level attacks. They do not prevent a legitimate researcher from retaining collected health data longer than promised, selling it without consent, or analyzing it for purposes beyond the original study. They do not address fundamental skepticism about data governance or institutional trustworthiness.

This mismatch between what TEEs protect and what users worry about explains the null effect on willingness and safety perceptions: even when users correctly understood TEEs, their adoption decision hinged on concerns beyond the technology's scope.

This study corroborates our thesis claim that education, while capable of improving comprehension, has fundamental limitations in driving adoption.

This work was submitted to the Journal of Cybersecurity in February 2026.

### *2.1.3 Formal Verification and Willingness to Adopt Password Managers*

Formal verification offers a mathematically rigorous basis for reasoning about software correctness . By proving that a program satisfies a formal specification, it eliminates entire classes of bugs with the same certainty as a mathematical theorem [24]. While experts in formal methods widely acknowledge its value for code quality and security assurance [24], end-user perceptions of formally verified software have received little attention prior to this work [10].

We chose password managers as a domain for investigating these questions. Password manager adoption is low despite expert recommendations [31], users report distrust and incomplete understanding as primary barriers [44], and recent verification efforts have been applied to password manager components such as the password generation algorithm [25]. This study addresses the gap by examining how knowledge of formal verification affects users' willingness to adopt a password manager and how users reason about the guarantees that verification provides.

The investigation proceeded in two phases. A formative qualitative study involved 15 participants recruited through the authors' network. We showed participants an extended version of the Bitwarden browser extension that incorporated a formal verification icon placed on verifiable features (the password vault, primary password input, password generator, and

**Table 2:** Percentage of participants that *agreed* or *strongly agreed* that the factor would impact their willingness to use a PM.

| Results | Factors |
|---|---|
| 69.0% | being inexpensive |
| 73.5% | having support materials (e.g., tutorials) |
| 76.5% | being certified by Password Manager Security Group |
| 78.0% | being free |
| 81.5% | being made by a trustworthy and familiar company |
| 86.5% | being mathematically correct |
| 87.0% | being easy to use for first-time/beginner users |

clipboard settings) and explanations developed iteratively with formal verification researchers with all technical jargon removed. Participants performed everyday password manager tasks and then completed semi-structured interviews. The formative study was exploratory: 90% of participants were unfamiliar with formal verification, and the primary goal was to surface initial reactions, misconceptions, and design requirements for the main study.

The main study was a 200-participant online survey conducted through Prolific. We decided to eliminate the term "formal verification" motivated by best practices for communicating with non-expert users [46]. Instead we told participants that the password manager is "mathematically correct, that is, its features are as trustworthy as a mathematical proof." This analogy follows the practice of media outlets that have used mathematical framings to explain formal verification to broad audiences [29, 47]. The survey assessed willingness to use the password manager relative to six other factors (including usability, certification, and cost) through five-point Likert scales. Two coders (Cohen's $\kappa = 0.81$) coded participants' open-ended responses using a 13 code codebook. Next, we presented seven scenario-based questions in which participants rated how likely they would be to stop using a password manager if a described failure occurred, addressing both users' feature priorities and their valuation of the guarantees verification provides.

**Willingness to Adopt.** The main study found that more than 86% of participants agreed or strongly agreed that a "mathematically correct" password manager would increase their willingness to use it. A Friedman test identified a statistically significant difference across the seven factor conditions ($\chi^2(6) = 51.42$, $p < 0.05$), but pairwise Wilcoxon comparisons did not find a significant difference between the formal verification factor and "easy to use" ($p > 0.05$), indicating that users value formal verification comparably to usability rather than as a uniquely compelling property. Formal verification and ease of use ranked at the top of the factor hierarchy, being inexpensive and having support features ranked at the bottom, with 31% and 27% of participants, respectively, reported that these latter factors would not make them more willing to use a password manager (see Table 2).

Participants' open-ended responses addressed why they valued, or did not value, formal

**Table 3:** Percentage of participants who *agreed* or *strongly agreed* that the described scenario would make them stop using a password manager. Scenarios are ordered by ascending impact.

| Results | Scenarios |
|---|---|
| 54.50% | S2 Policy Compliance |
| 70.50% | S8 Ransomware / Deleting your vault |
| 76.00% | S7 Synchronization |
| 79.00% | S5 Autofill |
| 83.00% | S6 Clipboard Clearing |
| 86.50% | S4 Primary Password Exposure |
| 92.50% | S1 Unpredictability |
| 96.00% | S3 Vault Exposure |

verification. The most frequent code was related to mathematics, 67 participants (35%) cited the mathematical nature of the guarantee as the primary reason for valuing it, with representative statements including *"mathematics doesn't lie"* (P89) and *"I like to use such products or services that have scientific proof of their effectiveness"* (P147).

More than 50% of participants referenced security as a reason for valuing formal verification. It is important to note that formal verification can ensure security properties, but it does not necessarily do so. As such, our results suggest that participants attributed to "mathematical correctness" a general assurance of security that exceeds what verification of specific components can guarantee. This pattern was already visible in the formative study, where 30% of participants concluded that the entire password manager was formally verified on the basis of seeing the verification icon in multiple locations, and where 5 of 10 participants described the icon as meaning simply that "passwords were safe" rather than identifying any specific verified property.

The scenario analysis shows that users prioritize vault security and primary password integrity above all other features: 96% agreed or strongly agreed they would stop using a password manager if vault exposure occurred (S3), and 92.5% gave the equivalent response for primary password exposure (S4). Scenarios involving probabilistic information leakage over time (password generator unpredictability, clipboard clearing, autofill) were impactful for more than 79% of participants. The feature that prior work has formally verified, password generator unpredictability [25], ranked among the lower-priority scenarios in users' expressed preferences (see Table 3).

Unlike the TEE study, where adoption was unaffected by the explanations, here, willingness to use password managers seemed to increase when formal verification was explained, suggesting that communication can influence willingness when the framing aligns with users' intuitive categories.

However, the same framing that makes formal verification accessible introduces a structurally different problem. Users over-attribute the scope of formal verification guarantees, extending the mathematical correctness framing to a general security guarantee that the tech-

nology does not provide.

The implication for the thesis is that intuition framing carries a dual risk. It succeeds as an adoption lever precisely because it exploits prior associations (e.g., mathematical rigor) that may not align with the guarantees formal verification provides. Responsible communication of formally verified software, therefore, requires not only framing that increases willingness, but explicit delineation of what formal verification does *not* cover, at the cost of potentially reducing the adoption benefit.

> **Carreira, C.**, Ferreira, J. F., Mendes, A., & Christin, N. (2025, November). Are Users More Willing to Use Formally Verified Password Managers?. In International Conference on Software Engineering and Formal Methods (pp. 185-202). Cham: Springer Nature Switzerland. [11]

**2.2 Developer Communication** The following studies examine developer communication in verification-aware programming, first by identifying the recurring difficulties practitioners face when interacting with these tools and then by evaluating whether LLMs can support that interaction in educational settings.

#### *2.2.1 Challenges Developers Face with Verification-Aware Languages*

Verification-aware (VA) programming languages, such as Dafny [36], Frama-C [20], Why3 [21], and Verus [35], extend conventional programming with constructs for formal specification and automated verification: preconditions, postconditions, and invariants that a verifier checks against user-defined contracts at compile time. These languages offer stronger correctness guarantees than testing alone and have been applied to industrial safety-critical systems [30, 42]. Despite their technical promise, adoption remains limited. Prior work attributes this to the expertise required from developers and a set of recurring practical obstacles, including unclear error messages [49], insufficient counter-examples for proof debugging [52], difficulty understanding why proofs fail [26], and the high cost of applying formal methods in development settings [55]. No prior study, however, had systematically collected and empirically validated the full landscape of challenges that VA language practitioners encounter across multiple tools.

The investigation used a three-step methodology. The first step was a structured literature review querying the ACM Digital Library, IEEE Xplore, Scopus, and Web of Science using a search string that combined formal verification terminology with the names of 14 VA languages. After screening 600 of the 1,024 retrieved articles on title, abstract, and full text, nine articles were identified as relevant. Their scarcity confirms both a lack of prior empirical work on the developer experience with VA languages and a set of recurring obstacle types that the subsequent steps were designed to validate and extend.

The second step extracted challenges from developer discussions on Stack Overflow, a platform whose question data has served as a source for empirical studies of developer difficulties across multiple programming paradigms [1, 6, 7, 27]. We collected 1,420 questions tagged with VA language names via the Stack Exchange API, the majority from Dafny (530) and Frama-C

(474). Following established pre-processing and model selection practices [3, 27], we trained a Latent Dirichlet Allocation (LDA) model [8] on a filtered dataset and identified eight topics, of which four yielded interpretable challenge categories: *Proof Elaboration*, *Environment Setup*, *Memory Model Understanding*, and *Language Understanding*.

The third step validated these challenge categories through an online survey of 31 practitioners recruited from formal methods courses, professional networks, GitHub contributors, and the Frama-C mailing list. Participants were divided by experience: 18 beginners (fewer than three years) and 13 experienced practitioners. The survey combined five-point Likert-scale questions on the challenges identified in the prior steps, two open-ended questions soliciting additional obstacles and improvement suggestions, and closed-ended questions rating specific proposed improvements. Quantitative data were analysed with Mann-Whitney U, Friedman, and Wilcoxon signed-rank tests; open-ended responses were coded inductively by two coders using an iterative codebook, yielding 14 codes for challenges and 13 for improvements.

**Adoption Challenges** The user study confirmed that the most prevalent challenges cluster around proof strategy and tool usability rather than environment configuration or language-specific syntax. Among the seven proof elaboration challenges evaluated, 64.5% of participants reported difficulty designing proof strategies accepted by the verifier ($PE_1$) and 48.4% reported difficulty completing designed proof strategies ($PE_2$). A Friedman test found a significant difference across the seven conditions ($\chi^2(6) = 27.79$, $p < 0.001$), and Wilcoxon pairwise comparisons identified $PE_1$ and $PE_2$ as significantly more prevalent than expressing termination clauses ($PE_7$, $p = 0.023$ and $p = 0.032$ respectively). The only challenge that differed significantly between beginners and experienced practitioners was expressing loop and recursion termination clauses ($p = 0.027$), which beginners found harder. This result indicates that the proof strategy barrier persists even as developers gain experience, pointing to a structural difficulty in the task rather than an entry-level knowledge gap that education alone can close.

Open-ended responses corroborated these findings. The most frequently reported categories were "Tool Performance Issues" and "Poor Error Messages and Feedback" (each cited seven times), followed by "Complex Language Features and Low-Level Operations" (six times) and "Specification and Annotation Challenges" (five times). These patterns echo findings from adjacent domains. Zhu et al. [56] found that Rust's safety features impose a steep learning curve and that compiler messages often lack actionable information. Christakis and Bird [17] documented that high false-positive rates and opaque feedback were primary reasons engineers at Microsoft rejected static analysis tools in practice. Ko et al. [34] identified understanding and information barriers as two of the six core barriers in programming systems. Overall, our results suggest that the principal obstacles to VA language adoption are not primarily syntactic familiarity but the cognitive overhead of proof construction and the inadequacy of tool feedback.

The context of practitioners' first contact with VA languages further illustrates the educational dimension of the barrier. Most beginners encountered VA languages through optional

university courses (89%), while most experienced practitioners were introduced to them in a work context (92%). As practitioners gained experience, self-teaching became increasingly prominent: 85% of experienced participants identified as self-taught. Formal education can provide initial exposure but may not supply sustained support that practitioners need to go beyond the introductory stage.

The most frequently requested improvements aligned with the identified barrier profile: better documentation and learning materials (five participants), improved usability (four), better error messages and feedback (three), and greater automation (three). Among the specific proposals evaluated, 83.9% of participants supported user-readable counter-examples to aid proof debugging, and 64.5% endorsed dedicated tutorials for common verification exercises. The finding that beginners placed significantly more emphasis on improved documentation search than experienced practitioners ($p = 0.028$) further supports the view that early-stage learners face distinct and unmet informational needs.

This study suggests that the main developer-side barriers to VA language adoption are the intrinsic difficulty of proof construction, insufficiently informative tool feedback, and a lack of educational resources. These findings reframe the education-adoption problem from the developer's perspective. Unlike the end-user studies, where comprehension improvements did not drive adoption because users' concerns lay outside the technology's scope, the present study establishes that comprehension and skill development are genuine and substantive barriers on the developer side: practitioners who cannot construct valid proof strategies or interpret verification failures cannot adopt VA languages regardless of their willingness to do so. Education is therefore necessary here, but it must be designed to address the specific obstacles identified, particularly the difficulty of developing proof intuition and obtaining actionable feedback from verification tools.

> Oliveira, F., Mendes, A., & **Carreira, C**. (2025, October). What Challenges Do Developers Face When Using Verification-Aware Programming Languages?. In 2025 IEEE 36th International Symposium on Software Reliability Engineering (ISSRE) (pp. 203-214). IEEE. [43]

### *2.2.2 LLMs as Educational Aids for Formal Verification*

Building directly on the barriers identified in Section 2.2.1, this study investigated whether large language models could serve as effective educational aids for students learning deductive program verification with Dafny [13]. Rather than evaluating LLM performance in isolation, the study situated LLM use within a controlled educational task, allowing us to examine how students interacted with the tool and how those interaction patterns mediated learning outcomes.

**Study Design.** We conducted a mixed methods study with 14 master's students enrolled in the elective course *Formal Methods for Critical Systems* at the Faculty of Engineering, University

of Porto, recruited after all lectures had concluded. Each participant solved two Dafny problems, a *Queue* problem based on a circular buffer and a *Tree* problem based on a binary search tree, one with access to a custom ChatGPT interface and one without. The interface replicated the standard ChatGPT experience while logging all interactions, enabling a direct link between prompting behavior and solution outcomes. Each problem comprised two subproblems: the first required participants to implement a method given its formal preconditions and postconditions, the second required them to both specify and implement a method from a natural language description alone. Participants were allotted 30 minutes per problem. Problem order and ChatGPT condition were balanced through bucketed randomization. The problems were designed from scratch and refined across five pilot sessions to be moderately challenging and to resist direct LLM solution, which proved to be a non-trivial design requirement in practice.

Data were collected from three sources: graded solution artefacts, the complete interaction logs (206 messages in total: 103 student prompts and 103 model responses), and a post-task survey covering self-assessed confidence, perceived usefulness, and trust. Correctness was evaluated on a 0–20 scale using a weighted, task-level rubric developed through an iterative double-blind coding process. Prompts and interaction sequences were categorised using an emergent codebook covering prompt characteristics, retry strategies, and overall interaction style.

**Performance.** LLM access produced a statistically significant improvement in student performance. The mean grade with ChatGPT was 17.39, compared to 9.36 without ($p = 0.0002$, $d = 1.55$, 95% CI: $[4.59, 11.47]$). All 14 participants achieved a passing grade when using the LLM, whereas only 5 of 14 did so unaided. The benefit was not uniform across task type. Performance on implementation subproblems (mean 19.52 with LLM, 9.90 without) substantially exceeded performance on specification subproblems (mean 15.26 with LLM, 8.82 without): a paired t-test on the combined subproblem showed that participants scored on average 2.92 points lower on specification tasks than on implementation tasks when using ChatGPT ($p = 0.003$, $d = 0.54$, 95% CI: $[-4.64, -1.19]$). This asymmetry is consistent with the relative scarcity of formally annotated code in LLM training corpora, which contain implementations far more frequently than formal contracts and invariants.

Despite strong aggregate gains, LLM-assisted performance did not uniformly translate into understanding or confidence. One participant scored zero unaided, producing no code and only explanatory comments, yet achieved a near-perfect score with ChatGPT while still reporting low confidence in the solution. This represented a disconnect between objective performance and self-assessed competence and suggests that a student can obtain a correct output through LLM interaction without acquiring the proof intuition or domain understanding that the task was designed to develop.

**Prompting Strategies and Successful Interaction.** Performance gains were strongly modulated by how students engaged with the LLM. We divided participants into three groups by their LLM-condition score: G1 (top five performers, scores 19 and 20), G2 (scores 17 and 18),

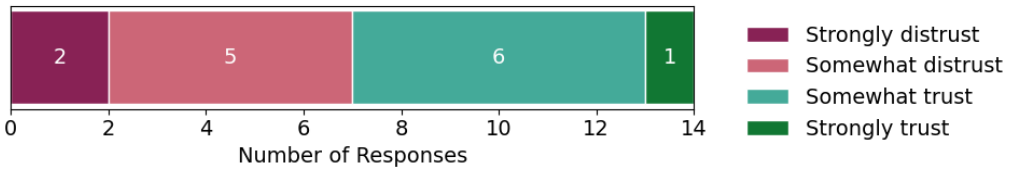


**Figure 1:** Distribution of self-reported trust in ChatGPT responses among the 14 participants in the LLM study. The near-equal split between trusting and distrusting participants underscores that trust is not uniformly conferred by LLM assistance and does not predict solution quality.

and G3 (lowest five performers, scores 10 through 16). Qualitative analysis of interaction logs revealed consistent differences. All five G1 participants provided the LLM with the full class definition in at least one prompt, supplying the contextual information necessary for the model to generate a correct solution, G3 participants who omitted this context forced the LLM to make structural assumptions that produced incorrect outputs. G1 participants also more frequently decomposed the task, directing each prompt to a single subproblem, a strategy observed in three G1 cases but only once in each of the other groups.

The most consequential differentiator between high and low performers was the degree to which students retained intellectual agency over the solution process. G1 participants were more likely to be classified as largely autonomous, making substantial manual modifications to LLM-generated code rather than accepting it verbatim. Two G1 participants also began with partial implementations before querying the LLM, a proactive structuring strategy absent from G3. By contrast, G3 participants more frequently attempted to redirect the LLM when it became stuck, a strategy that often induced progressively complex and ultimately incorrect responses rather than resolving the underlying verification difficulty. The most common repair technique across all participants was appending error messages or problematic code to subsequent prompts (43 prompts, 12 students), while unproductive redirection accounted for only 8 prompts. These patterns establish that better outcomes depended not on persistence or prompt volume but on the quality of contextual framing and the extent to which students exercised their own domain knowledge throughout the process.

**Trust.** Student attitudes toward the LLM were divided: seven participants reported trusting ChatGPT's responses and seven distrusted them. Among those who trusted the LLM, the most common reason was that it produced accurate code (three participants), while two noted that Dafny's built-in verifier allowed independent confirmation of the model's suggestions. Among those who distrusted it, the most frequent concern was the presence of syntactic errors in generated code (four participants), followed by the LLM's tendency to maintain incorrect claims with apparent confidence (two participants). Moreover, trust did not align straightforwardly with performance as participants who achieved high scores by passively accepting LLM output and those who engaged critically reported similar levels of trust. This suggests that trust is shaped by prior experience and domain confidence rather than by actual solution quality alone (see Figure 1).

**Implications.** These findings complement the analysis of Section 2.2.1. LLM assistance can substantially lower the performance floor for students who would otherwise be unable to begin a verification task, and this scaffolding function is non-trivial given the documented difficulty of proof strategy formulation identified in the prior study. However, the performance benefits are conditional on active, critically informed use: students who provided context rich and focused prompts and who modified the LLM's output engaged in the reasoning that formal verification requires, whereas those who deferred to the model verbatim obtained weaker results and did not develop the proof intuition that represents the principal long-term adoption barrier.

This implies that integrating LLMs into formal methods instruction is not sufficient on its own: the tool amplifies the capacity of students who already possess relevant domain knowledge, but does not substitute for its development. Educators must therefore teach productive prompting strategies alongside verification concepts, and must design assessments that make passive LLM reliance visible rather than inadvertently rewarding it. The result also suggests that unlike the end-user studies, where improving comprehension did not increase adoption because users' concerns fell outside the technology's scope, the developer studies establish that comprehension and skill development are genuine and addressable barriers, but that educational interventions must promote active learning rather than passive tool dependence to be effective.

> **Carreira, C.**, Silva, Á., Abreu, A., & Mendes, A. (2025, November). Can large language models help students prove software correctness? An experimental study with Dafny. In International Conference on Software Engineering and Formal Methods (pp. 203-220). Cham: Springer Nature Switzerland. [13]

## 3 Future Work

One additional study is proposed to resolve the open tension between adoption and accuracy in the communication of formally verified software.

**3.1 Explanations for Formally Verified Software** The two completed end-user studies establish, in isolation, the two principal failure modes of security technology communication. The TEE study (Section 2.1.2) suggests that explanations which accurately communicate a technology's guarantees can improve comprehension but not willingness to adopt. The formal verification explanation (Section 2.1.3) study suggests that a shallow, intuition-aligned explanation of formal verification increased willingness to adopt while making users over attribute the role of formal verification. The proposed study addresses the results of both previous studies, and I will investigate how to preserve the adoption benefit of intuition-aligned framing while attenuating the over-attribution cost.

**Research Questions.** The study is organized around three research questions.

**RQ1**: Can explanations improve comprehension of formal verification?

**RQ2**: Which components of an explanation of formal verification are most effective at improving adoption?

**RQ3**: Can explanations of formal verification reduce over-attribution, that is, the tendency to attribute guarantees beyond the scope of what has been verified?

**Explanation Conditions.** Following the compositional approach developed for the TEE study, candidate explanations are constructed additively so that each deeper condition is a strict extension of the shallower one. **E0** (control) informs participants that the software has been formally verified but provides no further characterization, this condition was absent from the willingness study and will serve as the reference against which all explanation conditions are compared. **E1** (shallow framing) replicates the "mathematically correct" condition from the willingness study, testing whether its adoption-positive result replicates in the new survey structure. **E2** (moderate explanation) identifies the specific components that have been verified and the properties those components are proven to satisfy, without explicit scope delineation. **E3** (detailed explanation with scope delineation) extends E2 with an explicit statement of what the verification does not cover: that the guarantee applies to the correctness of specific components against their specifications and does not imply that the broader system is verified, the specification itself captures all relevant threats, or that verified components cannot be deployed in a vulnerable context. This compositional structure directly operationalizes the Comprehension–Jargon and Awareness–Discomfort trade-offs identified in the systematic review (Section 2.1.1).

Prior to data collection, candidate explanations will be grounded empirically through a systematic analysis of formal verification explanations found in the wild, following the approach established in the TEE study. A structured web search targeting product marketing pages, academic press releases, and general-audience technology media will identify 30–50 explanations of formal verification produced for non-expert audiences. Two coders will independently apply an iterative codebook to identify recurring communication themes, including mathematical analogy, absence-of-bugs claims, component specificity, property specification, scope limitation, and trust transfer. Candidate explanations for each depth level will be constructed from themes with empirical precedent and verified for technical accuracy by formal verification researchers external to the team before deployment.

**Study Design.** Two between-subjects designs are under consideration. Under **Design A**, a mixed factorial structure is adopted: explanation condition (E0–E3) is a between-subjects factor and scenario is a within-subjects factor, with each participant completing four scenarios under their assigned explanation. This design supports direct tests of the Explanation × Scenario interaction, determining whether explanation effects on willingness and over-attribution are consistent across application domains or domain-specific. Under **Design B**, each participant completes exactly two scenarios: one under E0 and one under a randomly assigned treatment explanation (E1, E2, or E3). The within-person control comparison eliminates individual differences in prior familiarity and security attitudes from the explanation effect estimate. De-

sign A is better suited to isolating the contribution of individual explanation components and testing domain generalizability, Design B is better suited to establishing which explanations outperform no explanation as a practical baseline. The key question bearing on this choice is whether the Explanation × Scenario interaction is a primary or secondary scientific goal of the study.

**Scenarios.** Four scenarios are constructed, each describing a consumer facing context in which the participant encounters a formally verified software product. Scenario 1 (password manager) maintains continuity with the willingness study, Scenario 2 (secure messaging application) situates formal verification in the context of cryptographic protocol verification, testing whether adoption and comprehension patterns differ when the verified property is cryptographic rather than functional. Scenario 3 (AI-powered financial assistant) tests whether formal verification of an access control policy governing AI data processing is perceived as meaningful in a financial context, and whether it interacts with documented user skepticism toward AI in sensitive domains. Scenario 4 (medical data sharing application) enables an approximate cross-domain comparison with the medical scenario from the TEE study.

**Outcome Measures and Instrument.** The primary outcome measures are comprehension, willingness to adopt, perceived safety, and perceived guarantees. Comprehension is assessed through 10–12 true/false questions per scenario organized into two blocks. Block A covers the affirmative scope of the guarantee (what formal verification establishes in the given application), Block B covers the limits of the guarantee (what formal verification does not establish) and is the primary instrument for quantifying over-attribution. Items in Block B are constructed to distinguish two error types: *component-scope errors*, where participants attribute the guarantee to unverified parts of the same system, and *property-scope errors*, where participants attribute the guarantee to threat scenarios the verified property does not cover. This distinction directly operationalizes the over-attribution pattern documented in the willingness study, where participants extended a component-level guarantee to a general security assurance. Willingness to adopt is measured on a three-point scale consistent with the TEE study, perceived safety on a four-point scale following the design of the TEE study's second survey. Open-ended responses to "In your own words, what do you understand formal verification to guarantee about this application?" are double-coded by two coders using an iterative codebook targeting both over-attribution error types, following the qualitative procedure established in the TEE study.

**Recruitment and Analysis.** Participants will be recruited through Prolific with US-based inclusion criteria and gender quotas consistent with the TEE study. Sample size will be determined by a formal power analysis prior to data collection, targeting 80% power at $\alpha = 0.05$ for the between-subjects explanation main effect, using a planning estimate of $\eta_p^2 \approx 0.04$ consistent with the small to medium effects observed in the TEE study. Primary analysis of willingness and perceived safety will use the ART-ANOVA, which accommodates ordinal outcomes with-

out distributional assumptions and models both the Explanation main effect and the Explanation × Scenario interaction. Comprehension outcomes are analyzed using a Generalized Linear Mixed Model with a logit link, with by-participant random intercepts to account for repeated measures, the Block × Explanation interaction is the primary test of whether explanation conditions differentially improve awareness of non-guarantees relative to affirmative guarantees. Over-attribution rates from open-ended responses are compared across explanation conditions using a Kruskal-Wallis test with Dunn post-hoc comparisons using the Benjamini-Hochberg correction.

## 4 Timeline

The following schedule outlines the principal milestones from the thesis proposal presentation to the defense.

- **April 2026** Thesis proposal presentation.
- **May – June 2026** Future Study instrument development.
- **July – August 2026** Study data collection via Prolific and qualitative coding of open-ended responses by two independent coders.
- **September – October 2026** Study data analysis.
- **November – December 2026** Study paper writing and submission to a usable security venue.
- **January – April 2027** Thesis writing and revision.
- **June 2027** Thesis defense.

## 5 Conclusion

This thesis examines the relationship between education, comprehension, and adoption across three areas: end-user understanding of Trusted Execution Environments, user perception of formally verified software, and developer practice with verification-aware programming languages. The completed studies support a pattern in which explanation strategies improve comprehension but do not produce corresponding increases in willingness to adopt. The TEE study (Section 2.1.2) demonstrates that this decoupling arises when users' concerns fall outside the protection boundary of the technology; the formal verification willingness study (Section 2.1.3) identifies the converse, in which framing increases willingness to adopt while inducing over-attribution of the guarantee's scope. The programmer studies (Sections 2.2.1 and 2.2.2) establish that barriers to programming with verification-aware languages are addressable through educational intervention, but that effectiveness depends on engagement with the verification task rather than reliance on tool assistance. These findings support the thesis that education improves comprehension but is insufficient to drive the adoption of reliable security technologies, and motivate a future study (Section 3.1), which will determine whether explanation depth can preserve the adoption benefits observed under intuition-aligned framing while attenuating the over-attribution it induces.

## References


[1] Ahmad Abdellatif, Diego Costa, Khaled Badran, Rabe Abdalkareem, and Emad Shihab. Challenges in Chatbot Development: A Study of Stack Overflow Posts. In *MSR'20*, pages 174–185. ACM, 2020.

[2] Alessandro Acquisti, Laura Brandimarte, and George Loewenstein. Privacy and human behavior in the age of information. *Science*, 347(6221):509–514, 2015.

[3] Md Abdullah Al Alamin, Sanjay Malakar, Gias Uddin, Sadia Afroz, Tameem Bin Haider, and Anindya Iqbal. An empirical study of developer discussions on low-code software development challenges. In *2021 IEEE/ACM 18th MSR*, pages 46–57, 2021.

[4] Apple Inc. Differential privacy overview. https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf, 2017. Accessed: 2026-03-23.

[5] Gbadebo Ayoade, Vishal Karande, Latifur Khan, and Kevin Hamlen. Decentralized IoT data management using blockchain and trusted execution environment. In *Proceedings of the 2018 IEEE International Conference on Information Reuse and Integration*, 2018.

[6] Mehdi Bagherzadeh and Raffi Khatchadourian. Going big: a large-scale study on what big data developers ask. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, pages 432–442, New York, NY, USA, August 2019. ACM.

[7] Anton Barua, Stephen W. Thomas, and Ahmed E. Hassan. What are developers talking about? An analysis of topics and trends in Stack Overflow. *Empirical Software Engineering*, 19(3):619–654, June 2014.

[8] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[9] Cristian Bravo-Lillo. *Improving computer security dialogs: an exploration of attention and habituation*. PhD thesis, Carnegie Mellon University, 2014.

[10] Carolina Carreira, João F. Ferreira, Alexandra Mendes, and Nicolas Christin. Exploring usable security to improve the impact of formal verification: A research agenda. *First Workshop on Applicable Formal Methods (co-located with Formal Methods 2021).*, 2021.

[11] Carolina Carreira, João F Ferreira, Alexandra Mendes, and Nicolas Christin. Are users more willing to use formally verified password managers? In *International Conference on Software Engineering and Formal Methods*, pages 185–202. Springer, 2025.

[12] Carolina Carreira, Alexandra Mendes, João F Ferreira, and Nicolas Christin. A systematic review of security communication strategies: guidelines and open challenges. *arXiv preprint arXiv:2504.02109*, 2025.

[13] Carolina Carreira, Álvaro Silva, Alexandre Abreu, and Alexandra Mendes. Can large language models help students prove software correctness? An experimental study with Dafny. In *International Conference on Software Engineering and Formal Methods*, pages 203–220. Springer, 2025.

[14] Inmaculada Carrion Senor, José Luis Fernández-Alemán, and Ambrosio Toval. Are personal health records safe? a review of free web-accessible personal health record privacy policies. *Journal of Medical Internet Research*, 14(4):e114, 2012.

[15] Sunil Chaudhary, Tiina Schafeitel-Tähtinen, Marko Helenius, and Eleni Berki. Usability, security and trust in password managers: A quest for user-centric properties and features. *Computer Science Review*, 33:69–90, 2019.

[16] Yu Chen, Fang Luo, Tong Li, Tao Xiang, Zheli Liu, and Jin Li. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Information Sciences*, 522, 2020.

[17] Maria Christakis and Christian Bird. What developers want and need from program analysis: an empirical study. In *31st IEEE/ACM Int. Conf. on Automated Software Engineering*, pages 332–343, 2016.

[18] Andrey Chudnov, Nathan Collins, Byron Cook, Joey Dodds, Brian Huffman, Colm MacCárthaigh, Stephen Magill, Eric Mertens, Eric Mullen, Serdar Tasiran, et al. Continuous formal verification of Amazon s2n. In *International Conference on Computer Aided Verification*, pages 430–446. Springer, 2018.

[19] Rachel Cummings, Gabriel Kaptchuk, and Elissa M Redmiles. "I need a better description": An investigation into user expectations for differential privacy. In *ACM SIGSAC*, 2021.

[20] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-c: A software analysis perspective. pages 233–247, 2012.

[21] Jean-Christophe Filliâtre and Andrei Paskevich. Why3—where programs meet provers. In *Programming Languages and Systems: 22nd European Symposium on Programming 2013*, pages 125–128, 2013.

[22] Kelsey R Fulton, Anna Chan, Daniel Votipka, Michael Hicks, and Michelle L Mazurek. Benefits and drawbacks of adopting a secure programming language: Rust as a case study. In *17th Symposium on Usable Privacy and Security*, pages 597–616, 2021.

[23] Steven Furnell, Rawan Esmael, Weining Yang, Ninghui Li, et al. Enhancing security behaviour by supporting the user. *Computers & Security*, 75:1–9, 2018.

[24] Hubert Garavel, Maurice H ter Beek, and Jaco van de Pol. The 2020 expert survey on formal methods. In *Formal Methods for Industrial Critical Systems: 25th International Conference*, pages 3–69. Springer, 2020.

[25] Miguel Grilo, João Campos, João F. Ferreira, Alexandra Mendes, and José Bacelar Almeida. Verified password generation from password composition policies. In *17th International Conference on Integrated Formal Methods*, 2022.

[26] Matthias Güdemann. Online Teaching of Verification of C Programs in Applied Computer Science. In João F. Ferreira, Alexandra Mendes, and Claudio Menghi, editors, *Formal Methods Teaching*, 2021.

[27] Junxiao Han, Emad Shihab, Zhiyuan Wan, Shuiguang Deng, and Xin Xia. What do Programmers Discuss about Deep Learning Frameworks. *Empirical Software Engineering*, 25(4):2694–2747, July 2020.

[28] P. A. Hancock, A. D. Kaplan, K. R. MacArthur, and J. L. Szalma. How effective are warnings? a meta-analysis. *Safety Science*, 130:104876, 2020.

[29] Kevin Hartnett. Hacker-proof code. https://www.quantamagazine.org/formal-verification-creates-hacker-proof-code-20160920/, 2020. [Accessed 10-Jun-2025].

[30] Reiner Hähnle and Marieke Huisman. Deductive software verification: From pen-and-paper proofs to industrial tools. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10000:345 – 373, 2019.

[31] Iulia Ion, Rob Reeder, and Sunny Consolvo. ...no one can hack my mind: Comparing expert and non-expert security practices. In *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*, pages 327–346, 2015.

[32] Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. Why don't software developers use static analysis tools to find bugs? In *2013 35th ICSE*, pages 672–681. IEEE, 2013.

[33] Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W Reeder. A "nutrition label" for privacy. In *SOUPS*, pages 1–12, 2009.

[34] Amy J Ko, Brad A Myers, and Htet Htet Aung. Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages-Human Centric Computing*, pages 199–206. IEEE, 2004.

[35] Andrea Lattuada, Travis Hance, Jay Bosamiya, Matthias Brun, Chanhee Cho, Hayley LeBlanc, Pranav Srinivasan, Reto Achermann, Tej Chajed, Chris Hawblitzel, Jon Howell, Jay Lorch, Oded Padon, and Bryan Parno. Verus: A practical foundation for systems verification. In *SOSP*, pages 438–454, November 2024.

[36] K Rustan M Leino. Dafny: An automatic program verifier for functional correctness. In *International conference on logic for programming artificial intelligence and reasoning*, pages 348–370. Springer, 2010.

[37] Markus Lennartsson, Joakim Kävrestad, and Marcus Nohlberg. Exploring the meaning of "usable security". In *Human Aspects of Information Security and Assurance: 14th IFIP WG 11.12 International Symposium, HAISA 2020, Proceedings 14*, pages 247–258. Springer, 2020.

[38] Xavier Leroy, Sandrine Blazy, Daniel Kästner, Bernhard Schommer, Markus Pister, and Christian Ferdinand. Compcert - A formally verified optimizing compiler. In *ERTS 2016: Embedded Real Time Software and Systems, 8th European Congress*, 2016.

[39] McKenna McCall, Carolina Carreira, Miguel Flores, and Lorrie Faith Cranor. "You do understand that people don't trust technology?": Explaining trusted execution environments to non-experts. *Journal of Cybersecurity*, 2026. Submitted in February 2026; under review. McCall and Carreira contributed equally.

[40] Fan Mo, Zahra Tarkhani, and Hamed Haddadi. Machine learning with confidential computing: A systematization of knowledge. *ACM computing surveys*, 56(11), 2024.

[41] Pratik Musale and Adam J Lee. Trust TEE?: Exploring the impact of trusted execution environments on smart home privacy norms. *Proceedings on Privacy Enhancing Technologies*, 3, 2023.

[42] Mattias Nyberg, Dilian Gurov, Christian Lidström, Andreas Rasmusson, and Jonas Westman. Formal verification in automotive industry: Enablers and obstacles. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11247 LNCS:139 – 158, 2018.

[43] Francisco Oliveira, Alexandra Mendes, and Carolina Carreira. What challenges do developers face when using verification-aware programming languages? In *2025 IEEE 36th International Symposium on Software Reliability Engineering (ISSRE)*, pages 203–214. IEEE, 2025.

[44] S. Pearman, S. A. Zhang, L. Bauer, N. Christin, and L. F. Cranor. Why people (don't) use password managers effectively. In *SOUPS*, 2019.

[45] Fahimeh Raja, Kirstie Hawkey, Steven Hsu, Kai-Le Clement Wang, and Konstantin Beznosov. A brick wall, a locked door, and a bandit: a physical security metaphor for firewall warnings. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, pages 1–20, 2011.

[46] Elissa M Redmiles, Yasemin Acar, Sascha Fahl, and Michelle L Mazurek. A summary of survey methodology best practices for security and privacy researchers. Technical report, 2017.

[47] Paul Rubens. How playing computer games can make the world safer. https://www.bbc.com/news/business-33519194, Jul 2015. [Accessed 10-Jun-2025].

[48] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah. Trusted execution environment: What it is, and what it is not. In *Proceedings of the 2015 IEEE Trustcom/BigDataSE/Ispa*, 2015.

[49] Marc Schoolderman, Sjaak Smetsers, and Marko Van Eekelen. Is Deductive Program Verification Mature Enough to be Taught to Software Engineers? In *Proceedings of the 8th Computer Science Education Research Conference*, pages 50–57. ACM, 2019.

[50] Christian Stransky, Dominik Wermke, Johanna Schrader, Nicolas Huaman, Yasemin Acar, Anna Lena Fehlhaber, Miranda Wei, Blase Ur, and Sascha Fahl. On the limited impact of visualizing encryption: Perceptions of E2E messaging security. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, pages 437–454, 2021.

[51] Joshua Tan, Khanh Nguyen, Michael Theodorides, Heidi Negrón-Arroyo, Christopher Thompson, Serge Egelman, and David Wagner. The effect of developer-specified explanations for permission requests on smartphone user behavior. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 91–100, 2014.

[52] Vassil Todorov, Frederic Boulanger, and Safouan Taha. Formal verification of automotive embedded software. In *6th Conference on Formal Methods in Software Engineering (FormaliSE '18)*, page 84 – 87, 2018.

[53] Kapil Vaswani, Stavros Volos, Cédric Fournet, Antonio Nino Diaz, Ken Gordon, Balaji Vembu, Sam Webster, David Chisnall, Saurabh Kulkarni, Graham Cunningham, et al. Confidential computing within an AI accelerator. In *Proceedings of the 2023 USENIX Annual Technical Conference*, 2023.

[54] Yong Wang, June Li, Siyu Zhao, and Fajiang Yu. Hybridchain: A novel architecture for confidentiality-preserving and performant permissioned blockchain using trusted execution environment. *IEEE access*, 8, 2020.

[55] Neil White, Stuart Matthews, and Roderick Chapman. Formal verification: Will the seedling ever flower? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 375(2104), 2017.

[56] Shuofei Zhu, Ziyi Zhang, Boqin Qin, Aiping Xiong, and Linhai Song. Learning and programming challenges of Rust: A mixed-methods study. In *ICSE*, pages 1269–1281, 2022.