Summer of Bitcoin 2025

Project Proposal

Update SeedSigner BIP-85 implementation

Chaitanya Keyal

1. Introduction

Name: Chaitanya Keyal
Email: chaitanyakeyal@gmail.com
Phone: +91 91480 43218
Country: India
Timezone: IST (UTC +5:30)

University: Birla Institute of Technology and Science (BITS), Pilani
Majors: B.E. Computer Science, M.Sc. Mathematics

GitHub: Chaitanya-Keyal
LinkedIn: in/chaitanya-keyal
Telegram: okaybroo11
Discord: @okaybroo

I'm a second-year undergraduate student at BITS Pilani – Hyderabad Campus, pursuing a dual degree in B.E. Computer Science and M.Sc. Mathematics.

I primarily work with Python and Go. I enjoy solving low-level, systems-focused problems that require clean and maintainable solutions.

My interest in open source began early, and I've since found great satisfaction in contributing to meaningful, community-driven projects. I actively build and contribute to open-source projects as part of my university's programming club, where we collaborate on real-world problems.

I have experience working with a wide range of Python libraries, especially for backend development and command-line tooling. I'm comfortable working with unfamiliar codebases and place a strong emphasis on writing code that is modular, readable, and easy to maintain.

Outside of coding, I care deeply about good documentation, thoughtful design, and effective collaboration. I'm always eager to learn, contribute consistently, and grow as both a developer and a teammate.

2. Project Synopsis

SeedSigner currently supports BIP-85 child seed generation, but does not allow those derived keys to be loaded directly into the UI for use. This project proposes a cautious extension: allowing users to optionally load a derived child key into the interface for signing, address generation, and other actions as with any regular key. To minimize risk, the feature will be disabled by default and include clear UX warnings and visual indicators to distinguish them from primary seeds. Additional seed metadata will be surfaced in the UI to provide context, improving clarity without overwhelming the user. The goal is to make BIP-85 more practically useful—while keeping it safe.

3. Existing Contributions

I've made several contributions to SeedSigner across refactoring, new features, documentation, and code review. These have helped me develop a clear understanding of the codebase, its architecture, and the MVC structure it follows.

- PR #716: [Refactor] Replace manual brightness tip drawing with `ToastOverlay`:
  - Resolves a long-standing `TODO`
  - Refactored the `render_brightness_tip` method in `QRDisplayScreen` to use `ToastOverlay` instead of manually drawing elements.
  - Also extended `ToastOverlay` to support multiple lines of icon + text, customizable font, icon size, and per-icon vertical offsets—making it more modular and reusable.
  - This PR significantly cleaned up `QRDisplayScreen` and improved the modularity of the toast system.
- PR #722: [Feature] Warning Screen for High Tx Fees:
  - Resolves Issue #721
  - Added a warning screen that appears before the PSBT math screen if the transaction includes unusually high fees. The goal is to ensure the user is explicitly made aware of any excessive fees before proceeding to sign the transaction.
- PR #707: [Refactor] Enforce PEP-8 Compliance with Black:
  - Resolves Issue #715
  - Proposed introducing `black` as an automatic code formatter to enforce consistent code style across the project, aiming to reduce formatting-related discussions in PRs.
  - After discussing the idea with Keith Mukai and other contributors, the decision to implement this is currently deferred.
  - In addition to the PR, I thoroughly explored other available formatters, such as `yapf`, and looked into customizations to align them with the existing style guidelines. I documented my findings in this gist.
  - Although this process didn't immediately result in a decision to implement, it was an in-depth exploration of tools that could improve development efficiency and code quality.

- PR #703: [Documentation] Mention `--ignore-missing` flag issue on older macOS versions
    - Resolves Issue #507
    - Improved installation documentation by noting that the `--ignore-missing` flag of `shasum` is unsupported on macOS versions prior to Big Sur.
    - This update helps users avoid errors when using the unsupported flag on these versions.
- I've also participated in reviewing and providing feedback on the following PRs:
    - #699
    - #708
    - #719
    - #748

4. Project Overview

This project introduces an optional, safety-focused extension to the current BIP-85 implementation in SeedSigner to enable loading of child seeds while maintaining clear separation from primary seeds and avoiding dangerous patterns like recursive derivation.

This feature will build on the existing seed management framework with minimal disruption to the current architecture. Specifically:

- Update the BIP-85 derivation flow to optionally load a derived child seed immediately after selecting an index—removing the need to manually note down and re-enter seed words to use the child as a regular key, which can be error-prone and confusing.
- Introduce visual indicators and metadata for loaded child seeds (e.g., parent fingerprint, derivation index) on the Seed Options screen and/or TopNav, to ensure provenance transparency.

- Extend the settings menu to include three modes for BIP-85: Disabled (default), Enabled and Enabled with Loading, all gated behind explicit user selection and warning prompts.
- Optionally, block recursive derivation entirely at the UI and logic level to avoid confusing or dangerous nesting.
- Maintain full compatibility with existing Seed object interfaces to avoid widespread refactors.
- Update how in-memory seeds are rendered to distinguish loaded child seeds across all relevant screens.
- Add new flow tests for the BIP-85 child loading process and unit tests for affected logic.
- Update the screenshot generator to reflect changes to UI screens and metadata presentation.

By including clear visual indicators, this proposal aims to deliver a practical feature for advanced users while reinforcing SeedSigner's core values of simplicity, transparency, and caution.

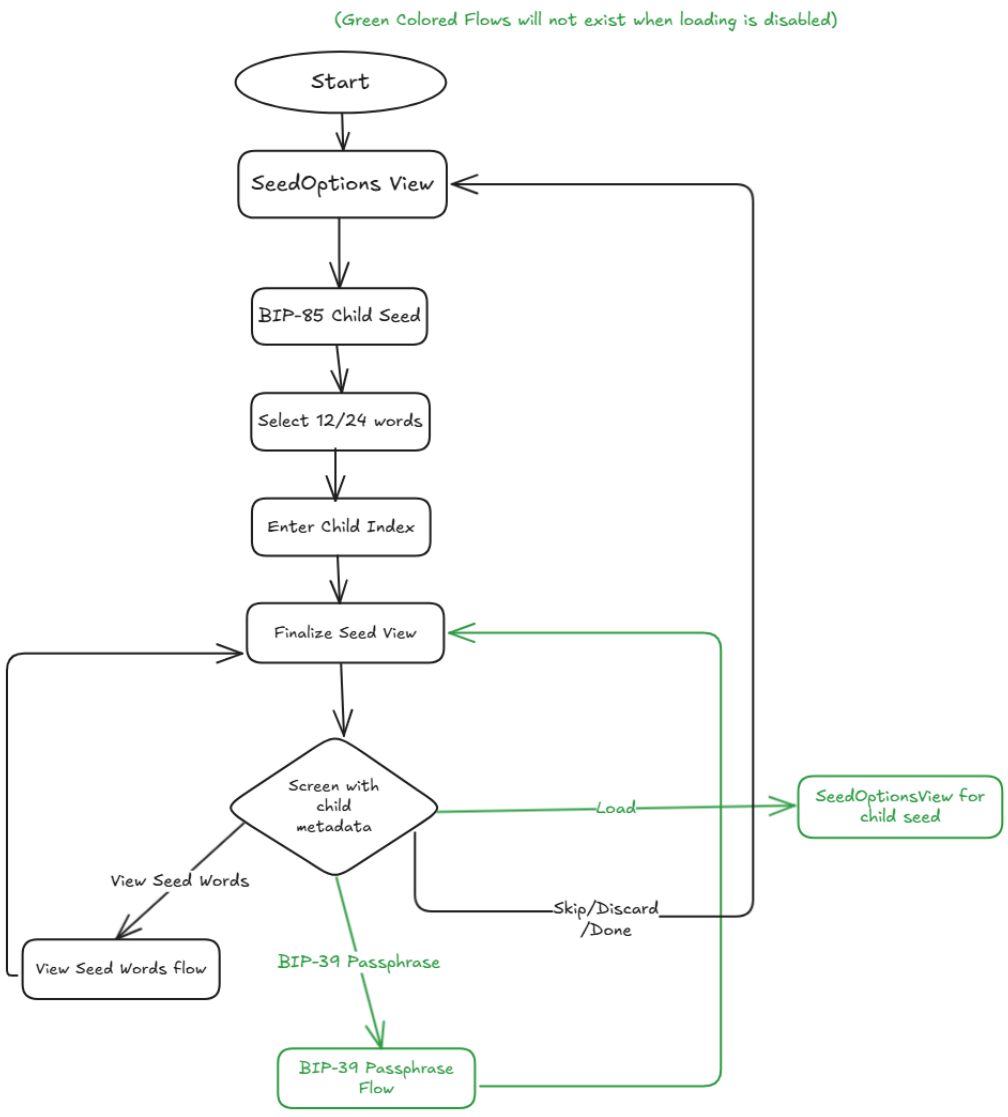
5. Implementation Details

Phase 1: Flow & UI/UX Finalization

Tasks:

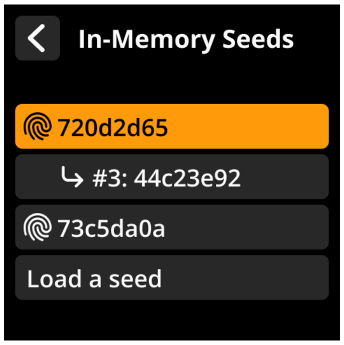
- Select, discuss and finalize the most suitable BIP-85 loading flow:
  - A few different options are outlined in this Excalidraw board, each exploring various trade-offs between user flexibility, clarity, and safety.
  - The flow I believe best balances those factors is outlined below:
    - Always display the child key's fingerprint, index, and word count.
    - Providing an option to directly load the child seed into memory without requiring users to view or back up the seed

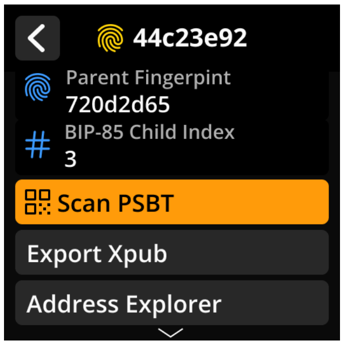
words—leveraging BIP-85's core advantage of deterministic derivation without additional backup overhead.
    - Still including a "View Words" option for users who prefer to inspect the seed before loading or for external use.

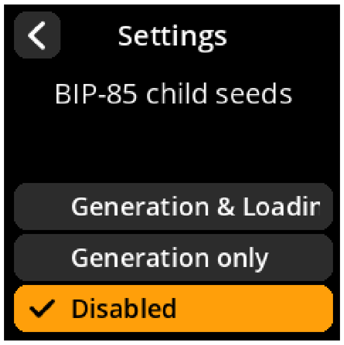


- Create UI mockups for:
    - SeedOptions screen: Show child seed metadata (e.g., parent fingerprint, derivation index)
    - TopNav or active seed indicator: Subtle visual cue (e.g., child icon or color code)

- In-memory seed list: Clearly distinguish child seeds and possibly show relationship with parent
- Settings menu: BIP-85 modes (Disabled, Enabled, Enabled with Loading) with appropriate warnings
- Confirmation/warning screens: Prompt user before loading a child seed to ensure awareness.
- Finalize Seed screen: Add metadata such as parent fingerprint, child index, and number of words.







- Gather feedback from the community to refine the flow, layouts, warning messages, and settings structure.

Deliverable: Final flowchart and UI mockups guiding implementation.

Phase 2: Core Derivation & Load Flow

Tasks:

- Implement the finalized BIP-85 flow.
- Add a method to load a derived child seed into in-memory storage to be used like any other seed.
- Update the Seed class to store BIP-85-related properties such as the parent fingerprint and derivation index.

Deliverable: Functional “Load BIP-85 Child” feature in the codebase.

Phase 3: Visual Indicators & Metadata

Tasks:

- Update SeedOptions screen to display metadata for child seeds:
    - Parent fingerprint
    - BIP-85 index used
- Add visual indicators (e.g., icon or alternate color) to distinguish child seeds at a glance.

Deliverable: Non-intrusive visual cues that make child provenance immediately obvious.

Phase 4: Seed List Rendering Updates

Tasks:

- Modify rendering logic to distinguish child and parent seeds within in-memory displays.
- Consistently apply the selected visual style (e.g., icons, suffixes) across all relevant screens.

Deliverable: Uniform, intuitive display of child and parent seeds that integrates seamlessly with existing workflows.

Phase 5: Settings & Safeguards

Tasks:

- Add a new BIP-85 Mode setting with three options:
    - Disabled (default)
    - Enabled (generation only)
    - Enabled with Loading
- Consult with the community regarding nested child derivation:
    - If disallowed, block recursive derivation both in the UI and logic layer.

    - If allowed, update in-memory seed display to reflect nesting visually (e.g., indentation).

Deliverable: Settings infrastructure that gives users fine-grained control over BIP-85 behavior.

Phase 6: Tests, Screenshot Generator & Documentation

Tasks:

- Write integration and flow tests for child seed loading.
- Create unit tests for core methods supporting child seed handling.
- Update the screenshot generator to include new UI states and metadata indicators.
- Add translator notes for all new messages, warnings, and labels.
- Expand documentation to cover all new functionality.

Deliverable: Fully tested and documented feature set.

Phase 7: Bug Fixes, Final Polish & Buffer

Tasks:

- Address feedback from mentors and the community.
- Fix bugs discovered during integration and testing.
- Perform layout and rendering consistency checks across devices and locales.
- Conduct a final review of code quality, naming conventions, and UI styling.

Deliverable: Production-ready, polished codebase with stabilized BIP-85 child-loading feature.

6. Project Timeline

Prior Commitments
I have no major summer obligations aside from university exams, which end on May 15—just before the coding period begins. I will be on a short, off-grid trekking trip from May 18–22; to accommodate those four days, I plan to allocate additional time on other days that week to keep the project on schedule.

May 8 – May 15 (Program Kick-Off and Onboarding)
Until now, I've been working with the SeedSigner emulator. However, testing on actual hardware is essential. During this period, I plan to acquire the necessary components and set up a local manual build on my own SeedSigner device. I will also familiarize myself with development on the Raspberry Pi and discuss the proposed BIP-85 flows with the community.

May 16 – May 31

- Finalize and approve the BIP-85 load flowchart
- Complete and vet all UI mockups with community feedback

June 1 - June 21

- Implement the "Load BIP-85 Child" logic
- Extend Seed class to carry BIP-85 metadata (parent fingerprint, derivation index)

June 22 – June 30:

- Update SeedOptions, TopNav, seed lists with child-seed indicators
- Ensure rendering logic distinguishes parent/child seeds

July 1 - July 5 (Mid-term Evaluations)

- Functional BIP-85 Child loading Flow.
- UI Changes to distinguish child seeds.

July 6 – July 21

- Update BIP-85 settings options and warning dialogs
- Based on community decision, either block nested derivation or update UI to display nested seeds clearly

July 22 – August 4

- Write integration/unit tests for child loading
- Update screenshot generator and all docs/I18n strings

August 5 – August 15 (Final Weeks)

- Address feedback from mentors and community reviewers.
- Ensure consistent visuals and behavior across different locales and devices.
- Buffer time for any final adjustments or bug fixes.

7. Future Deliverables

Post-SoB, I'm keen to continue contributing to SeedSigner. I love the idea of having my own inexpensive, air-gapped Bitcoin signing device, and collaborating with other contributors has already been an invaluable experience. I'm confident it'll only get better over the summer. Some specific areas I'd be excited to explore after the SoB period include:

Improving Developer Tooling: Building better tooling for automated screenshot generation, setting up GitHub Actions to run tests, streamlining the translation workflow, and continuing discussions around adopting formatters and linters to keep the codebase clean and consistent.

Touchscreen Support: This is something that really intrigues me. Touchscreen support would be a major feature upgrade for SeedSigner, and I'd love to contribute to enhancing the UI for touch input and dive deeper into understanding the hardware and integration needed to support it.

Ultimately, I'd like to stick around as a long-term contributor—focusing not just on building new features, but also on reducing friction for both users and developers as the project evolves.

8. Benefits to Community

This enhancement gives advanced users a streamlined, integrated way to work with BIP-85–derived child seeds directly in SeedSigner—no more error-prone manual re-entry or external tooling. At the same time, clear visual cues, warning prompts, and a default-off setting ensure that casual users remain protected from the complexity and risks of deterministic key derivation.

Beyond enabling direct loading, the UI will surface child-seed provenance at every step—displaying the child's fingerprint and index during derivation, and showing metadata (parent fingerprint and derivation index) in the SeedOptions view. This transparency helps users verify exactly which key they're using—avoiding the "child-of-child-of-child" trap—and reinforces trust in SeedSigner's secure, minimalist design.

9. Past Projects

As mentioned earlier, I've actively contributed to and built several open-source projects. Some notable ones include:

- Arcade:
    - Online multiplayer games (Chess, Monopoly) built from scratch using `tkinter`.
    - Features include real-time token movement, smooth game animations, and an intuitive GUI for a seamless player experience.

- Search Service for Chronofactorem:
    - Chronofactorem helps students generate semester timetables and detect exam/lecture clashes.
    - I built an Elasticsearch-powered search service for fast, structured retrieval of university courses and personalized timetables.
- Chrono2GCal:
    - A tool that integrates Chronofactorem timetables directly with Google Calendar.
    - It automatically adds classes and exams with details like location and instructor, and supports customizations for event colors, reminders, and titles.
- BitsGPT (Rewrite):
    - BitsGPT is an agentic chatbot tailored for BITS students, answering questions related to academics, placements, and campus life. I rewrote the backend to improve modularity, accuracy, and performance, enabling more reliable and maintainable responses.
- onlineCAL:
    - A Django-based portal for the Central Analytical Laboratory's instrument booking system, designed to streamline slot reservations and administrative control.
- *async*hrony:
    - An LLM-powered personal agent that can dynamically generate queries for accurate personal financial data retrieval, efficiently classify user requests, and handle complaints in real-time, ensuring instant and precise responses.
- BioHue:
    - A health-tech app that analyzes color changes on specialized bandages to detect infection probabilities.