Würfel-Check – hat der SeedSigner richtig gerechnet?

Drei Wege, den Dice-Modus unabhängig nachzurechnen · gehört zur SeedSigner-Anleitung 1b und zum Würfelprotokoll-Blatt 1b · Kapitel 3

⚠ **Geprüft wird die Methode, nie der echte Seed.** Die Würfelziffern sind der Seed – wer sie in einen Rechner tippt, hat den Seed in einen Rechner getippt. Deshalb: Einen **Wegwerf-Seed** würfeln, den nachrechnen, dann verwerfen. Rechnet das Gerät bei Übungsziffern richtig, rechnet es immer richtig. Der produktive Seed entsteht danach am Gerät und verlässt es nie.

⚠ **Auch das Prüfwerkzeug ist Angriffsfläche.** Eine manipulierte Prüfseite könnte „stimmt" sagen, wo nichts stimmt – oder Ziffern mitlesen. Darum: Dateien vor der Nutzung gegen die Prüfsummen unten abgleichen, offline arbeiten, und wenn es ernst ist, zwei unabhängige Werkzeuge gegeneinander laufen lassen.

WAS DU BRAUCHST

- ☐ Einen Wegwerf-Durchlauf am SeedSigner: **Ziffern beim Eintippen mitschreiben** (Würfelprotokoll-Blatt 1b) und die angezeigten Wörter notieren – bei einem Übungs-Seed ist das erlaubt.
- ☐ Aus dem Werkstatt-Paket: `wuerfel-check.html` (Weg 1) und/oder `verify_dice_seed.py` + `bip39_english.txt` + `SHA256SUMS` + `dice-check.sh` (Weg 2). Optional Ian Colemans `bip39-standalone.html` (Weg 3).
- ☐ Ein Rechner, den du **vom Netz trennen** kannst – oder ein Tails-Stick (Kür, nicht Pflicht).

WEG 1 – WÜRFEL-CHECK IM BROWSER (DER EINFACHSTE)

Eine einzelne HTML-Datei, kein Server, keine Verbindung. Rechnet SHA-256 mit zwei Rechenwerken (Browser-WebCrypto + eigene Implementierung), prüft die eingebaute Wortliste per Hash und läuft beim Öffnen einen Selbsttest gegen die Beispiele der SeedSigner-Dokumentation.

1. ☐ Datei von der Download-Seite speichern (Rechtsklick → „Ziel speichern unter"). **Prüfsumme vergleichen** (unten), Wert stimmt: _____
2. ☐ **Netz trennen** (WLAN aus, Kabel raus). Datei per Doppelklick öffnen – sie läuft aus dem Dateisystem, der Browser darf laut Content-Security-Policy nichts nachladen.
3. ☐ Statuszeile lesen: **Verbindung offline · Wortliste OK · SHA-256 stimmt überein · Selbsttest OK**. Ist eine der vier Anzeigen rot, Datei nicht verwenden.
4. ☐ Ziffern eingeben (99 für 24 Wörter, 50 für 12; Leerzeichen erlaubt), darunter optional die Wörter vom Display, dann **„Nachrechnen"**. Grün = alle Wörter identisch. Rot = die abweichenden Wörter sind markiert.
5. ☐ **„Alles löschen"**, Browserfenster schließen. Nachprüfen für Skeptiker: F12 → Netzwerk-Tab zeigt keine einzige Anfrage, auch nicht beim Rechnen.

WEG 2 – PRÜFSKRIPT IM TERMINAL (LINUX, MACOS, TAILS)

Rund 90 Zeilen Python, nur Standardbibliothek plus die offizielle Wortliste. Der Wrapper `dice-check.sh` prüft vor jedem Start die Hashes beider Dateien gegen `SHA256SUMS` und startet erst dann.

1. ☐ Die vier Dateien in einen Ordner legen, Hashes gegen die Tabelle unten prüfen: `sha256sum -c SHA256SUMS` → zweimal „OK". Auf Tails: Ordner in der Persistenz anlegen, damit die Dateien den Neustart überleben.
2. ☐ Selbsttest: `bash dice-check.sh --selftest` → „Selbsttest 99 Würfe / 24 Wörter: OK" und „… 50 Würfe / 12 Wörter: OK".
3. ☐ Netz trennen. `bash dice-check.sh` **ohne Argument** – das Skript fragt die Ziffern ab. Nie als Argument übergeben, sonst stehen sie in der Shell-History.
4. ☐ Die ausgegebenen Wörter mit dem Display vergleichen („Seed Words: 1/6" bis „6/6"). Danach `clear` oder Terminal schließen – die Wörter stehen sonst im Scrollback.

Weg 3 – zweite Codebasis (Ian Coleman): `bip39-standalone.html` vom offiziellen Release laden, offline öffnen → „Show entropy details" → **„Base 10"** aktiv anklicken → **„Mnemonic Length" auf „24 Words"** (bzw. „12 Words"; bleibt „Use Raw Entropy" stehen, wird nicht gehasht und die Wörter weichen ab) → Ziffern eintragen. **Nie den „Dice"-Modus** (macht aus jeder 6 eine 0). Der SeedSigner hat Colemans „Base 10"-Rechenweg übernommen (Ziffern als Text, SHA-256), deshalb passt er zum Gerät.

Warum drei Wege? Ein Werkzeug kann sich irren oder manipuliert sein. Zwei unabhängige Codebasen (Browser-JavaScript und Python, oder eines davon plus Coleman), die auf dieselben Wörter kommen wie das Gerät, sind ein Beweis. Ein Werkzeug allein ist ein Indiz. **Beweist nicht:** ob deine Würfel fair sind und ob dein Backup lesbar ist – dafür gibt es Würfelbecher und Restore-Drill.

PRÜFSUMMEN (SHA-256) – GEGEN DIE DOWNLOAD-SEITE ABGLEICHEN

`bip39_english.txt`	`2f5eed53a4727b4bf8880d8f3f199efc90e58503646d9ff8eff3a2ed3b24dbda (offizielle BIP39-Liste, identisch mit dem Bitcoin-BIPs-Repository)`
`verify_dice_seed.py`	`96bcf6a2062f5035ff9a3f2eeda455122db7f899d77a7d461f2f78c5e81f9b27`
`wuerfel-check.html`	`siehe Download-Seite (die Datei kann ihren eigenen Hash nicht enthalten) – Wert hier eintragen: _____`
Prüfbefehl	`Linux/Tails/macOS: sha256sum DATEI · Windows PowerShell: Get-FileHash DATEI`

ERGEBNIS

- ☐ **Bestanden:** Werkzeug(e) und Gerät liefern dieselben Wörter → der SeedSigner rechnet standardkonform. Übungs-Seed verwerfen, echten Seed am Gerät würfeln, Ziffern nirgendwo mehr eintippen.
- ☐ **Abweichung:** Häufigster Grund ist ein Tippfehler zwischen Blatt und Gerät. Ziffern prüfen, Durchlauf wiederholen. Weichen zwei Werkzeuge einhellig vom Gerät ab: Gerät nicht verwenden, Firmware-Version und Bezugsquelle prüfen.