STABLENOTES V1 WHITEPAPER

V 1.0.0

Non-Custodial Cold Storage & Durable Recovery Media for Tokenized Assets

By Alex Breton
alex.breton@stablenotes.org

LEGAL DISCLAIMER

This Whitepaper (including any appendices, exhibits, linked materials, updates, and revisions, the "Whitepaper") is provided by PrintDreams International AB and/or its affiliates ("PrintDreams," "we," "us," or "our") solely for informational and discussion purposes in relation to Stablenotes v1 and related software development kits (SDKs), reference implementations, and associated devices and media (collectively, the "Technology"). This Whitepaper is not intended to constitute, and does not constitute legal, tax, accounting, financial, or investment advice. You should obtain independent professional advice before acting on the basis of this Whitepaper.

No offer or solicitation; EU financial promotion and prospectus disclaimer
This Whitepaper does not constitute, and shall not be construed as, an offer to sell or a solicitation of an offer to buy any securities, transferable securities, financial instruments, crypto-assets, tokens, derivatives, or investment products in any jurisdiction, including within the European Economic Area ("EEA"). No prospectus has been approved or will be approved for the purposes of Regulation (EU) 2017/1129 (the "Prospectus Regulation") in connection with this Whitepaper, and no such prospectus is intended to be published. Any offer or distribution, if ever made, would be made only pursuant to definitive documentation and in compliance with applicable laws and regulatory requirements.

MiCA / CASP disclaimer; Technology is not a regulated crypto-asset service
Nothing in this Whitepaper is intended to constitute an offer to the public of crypto-assets, an admission to trading of crypto-assets, or the provision of crypto-asset services within the meaning of Regulation (EU) 2023/1114 (Markets in Crypto-Assets Regulation, "MiCA"). PrintDreams does not, by publishing this Whitepaper or providing the Technology, represent that it is authorized as a crypto-asset service provider ("CASP") or that the Technology is offered as a regulated crypto-asset service. Any references to blockchain networks, crypto-assets, stablecoins, wallets, or smart contracts are for technical description only.

Self-custody; no custody or control of user crypto-assets
Stablenotes v1 is designed as a self-custodial technology. Users are solely responsible for safeguarding their private keys, recovery materials, devices, and credentials. PrintDreams does not take custody of, control, access, or manage user crypto-assets, private keys, or wallets and cannot recover lost keys or reverse blockchain transactions. Blockchain transactions are generally irreversible. Loss, theft, damage, compromise, or unavailability of private keys or recovery materials may result in permanent loss of crypto-assets.

Third-party services (including regulated providers)
The Technology may be used in conjunction with third-party products and services, including but not limited to blockchain networks, wallets, smart contracts, decentralized applications, crypto on-/off-ramp providers, payment card programs, and other service providers ("Third-Party Services"). Third-Party Services are not controlled by PrintDreams and may be subject to separate terms, fees, eligibility requirements (including KYC/AML and sanctions screening), and regulatory authorizations. PrintDreams does not provide, and is not responsible for, Third-Party Services. Where Third-Party Services provide regulated activities (including crypto-asset services, payment services, or e-money services), such activities are provided solely by those third parties and not by PrintDreams.

PSD2 / payments disclaimer; not a payment service provider
PrintDreams is not authorized as a payment institution or electronic money institution and does not provide payment services within the meaning of Directive (EU) 2015/2366 ("PSD2") or issue electronic money. Any payment or card-related functionality described in connection with the Technology, if any, would be provided by Third-Party Services subject to their own regulatory status and terms.

Regulatory uncertainty and geographic restrictions
Laws and regulations applicable to crypto-assets, digital finance, and related technologies continue to evolve and may differ between EU Member States and other jurisdictions. The availability or functionality of the Technology may be restricted, modified, geofenced, suspended, or discontinued in certain jurisdictions to comply with applicable law or risk controls. It is your responsibility to ensure compliance with applicable laws and regulatory requirements in your jurisdiction.

Data protection (GDPR) and privacy
To the extent PrintDreams processes personal data in connection with the Technology, such processing will be governed by applicable data protection laws, including Regulation (EU) 2016/679 (the "GDPR") and any applicable Member State implementing laws. This Whitepaper does not constitute a privacy notice. Any privacy notices, data processing terms, and related disclosures will be provided separately. Where Third-Party Services process personal data (including for KYC/AML), such processing is governed by the third party's privacy documentation and legal obligations.

Security, operational resilience, and reliability risks
Use of the Technology involves material risks, including software defects, smart contract vulnerabilities, cybersecurity incidents, supply-chain compromise, device failures, printing/media degradation, tamper/counterfeit attempts, data loss, and service interruptions. PrintDreams does not guarantee that the Technology will be secure, uninterrupted, error-free, or immune from attack. Any security, durability, or survivability targets stated in this Whitepaper are objectives only and may depend on environmental conditions, user handling, third-party components, and operational controls. Nothing in this Whitepaper should be interpreted as a representation that the Technology meets any specific regulatory technical standard (including under DORA, NIS2, or other ICT/security frameworks) unless expressly stated in writing by PrintDreams.

Intellectual property; no license granted except as stated
All intellectual property rights in and to the Whitepaper and the Technology are owned by PrintDreams or its licensors. Except as expressly set out in a written agreement with PrintDreams, no license or rights are granted under any patent, copyright, trademark, trade secret, or other intellectual property right by implication, estoppel, or otherwise.

Limitation of liability
To the maximum extent permitted by applicable law, PrintDreams shall not be liable for any direct, indirect, incidental, special, consequential, exemplary, or punitive damages, or any loss of profits, revenue, goodwill, data, or crypto-assets, arising out of or related to this Whitepaper or the use of (or inability to use) the Technology, whether in contract, tort, strict liability, or otherwise, even if advised of the possibility of such damages. Nothing in this disclaimer limits liability to the extent such liability cannot be excluded or limited under applicable law.

No reliance; forward-looking statements
The Whitepaper may contain forward-looking statements based on current expectations and assumptions. These statements are inherently uncertain and may differ materially from actual outcomes. By accessing or using this Whitepaper, you acknowledge that you have not relied on any statement or representation not expressly set out in definitive written agreements with PrintDreams.

Updates
PrintDreams may revise, update, or replace this Whitepaper at any time without notice. The version and date stated on the cover page apply only to that version of the Whitepaper.

EXECUTIVE SUMMARY

Stablenotes v1 is an SDK-first recovery layer for self-custodial crypto and stablecoin wallets. It enables wallet providers to offer users a durable, offline recovery instrument—a printed "note" containing an encoded recovery payload (e.g., QR + human-readable fallback) produced through a certified printing device workflow and governed by clear security and reliability requirements.

What v1 is

Stablenotes v1 is not a new blockchain, not a custodian, and not a payments network. It is a set of components that can be integrated into existing wallets and apps:

- **Stablenotes SDK:** APIs and reference flows that wallet apps can integrate to create, verify, and recover from a printed recovery instrument.
- **Recovery media specification:** a defined format for the printed payload (QR + fallback) and durability/tamper-evidence requirements.
- **Mobile ATM Device:** a certified, personal, handheld printing device that prints the payload on top of the recovery media.
- **Device integration kit:** a certified printing workflow and quality gates that enforce post-print verification and prevent unsafe issuance.

The v1 product objective is narrow: help users retain access and protect their tokenized assets even when phones are lost, compromised, or unavailable, without requiring custody by a third party.

Why v1 matters

Self-custody is increasingly common in crypto and stablecoin markets, particularly in environments where users want direct control and 24/7 availability. However, self-custody fails disproportionately due to avoidable issues:

- lost or compromised phones,
- hacking attempts,
- fragile backup practices,
- user error during recovery,
- and low resilience in intermittent connectivity or high-risk environments.

Stablenotes v1 addresses this gap by providing a physical, user-controlled recovery mechanism that can be created and verified through a standardized, auditable workflow—delivered as infrastructure that existing wallets can embed.

How v1 works (high level)

At a high level, a partner wallet integrates the Stablenotes SDK to support a controlled "Create Recovery Note" workflow:

1. User initiates recovery-note creation within a self-custodial wallet.

2. The wallet generates the relevant recovery material using platform cryptographic randomness (CSPRNG) and applies strict non-retention and memory hygiene controls.
3. The recovery payload is encoded into:

- a machine-readable QR, and
- a human-readable fallback with checksum/integrity markers.

4. The payload is printed onto qualified media using a controlled printing workflow.
5. A post-print verification gate requires scan and integrity checks before finalizing the issuance step.
6. The user stores the printed recovery instrument offline.
7. If recovery is needed, the user imports/scans the recovery instrument to restore access according to the partner wallet's recovery flow and the v1 contract/spec rules.

Stablenotes v1 is designed to scale through partner integrations, not through building a proprietary consumer financial network.

Non-goals (explicit)

To avoid expanding regulatory and security scope, Stablenotes v1 explicitly does not include:

No bearer transfer: v1 is not designed or marketed as a transferable "cash-like" bearer instrument.

No cash-out network: v1 does not include agent/merchant cash-out distribution or redemption rails.

No operator program: v1 does not subsidize operators, deploy agent networks, or run retail payout infrastructure.

No token sale or tokenomics: v1 includes no token issuance, no revenue-share token, and no token fundraising mechanics.

Any "bearer transfer" or offline payments use case is outside v1 scope and would require separate technical, regulatory, and adversarial validation.

What we have proven

SDK beta exists and supports core lifecycle flows in a controlled environment.

Reference implementation wallet apps are available on Android and iOS.

Mobile ATM device is certified in several large markets (EU, USA, Canada, UAE, India) and in mass production. First batch has already been manufactured, and the technical platform is well proven.

What remains to be proven

Device printing workflow is certified and supports controlled printing and verification gates.

Initial security controls implemented: RNG usage, non-retention design, and SDLC practices.

Initial durability/tamper baseline testing to be completed on qualified media.

Independent audits covering RNG/key generation, non-retention, firmware/update integrity, and smart contracts.

Independent durability qualification with defined service-life targets and acceptance thresholds across representative device matrices.

Operationalized supply chain controls for serialized media (chain-of-custody, quarantine/recall drills).

Partner integration certification program (security boundary verification, telemetry controls, incident runbooks) and successful production deployments with design partners.

Field evidence on user handling, support burden, and failure rates at meaningful scale.

0 DOCUMENT INFORMATION

Purpose: Provide document control and revision history suitable for partner/legal diligence.

0.1 Document metadata

- Document title: **Stablenotes v1 Whitepaper**
- Document ID: **SN-WP-V1-YYYY-[MM]**
- Version: **v[MAJOR].[MINOR].[PATCH]**
- Status: **Final Draft**
- Owner: **A. Breton, CEO**
- Reviewed by: **M. Kosenko, Analyst**
- Approved by: **A. Ricknell, Director**
- Effective date: **2026-03-19**

0.2 Versioning policy

- **MAJOR**: changes to scope, security claims, trust boundaries, or any Go/No-Go gate
- **MINOR**: additions/clarifications that do not change claims or gates
- **PATCH**: typos, formatting, non-substantive edits

0.3 Change log table

Version	Date	Author	Summary of changes	Sections Affected	Evidence impact	Approval
v 0.1.0	2025-08-21	A. Breton	Initial draft structure	All	N/A	M. Kosenko
v 1.0.0	2026-03-19	A. Breton	v1 scope lock	All	v1 and v2 separation	A. Ricknell

Evidence Impact guidance (required column):

- “None”
- “New evidence required”
- “Existing evidence invalidated”
- “Protocol updated — re-test required”

TABLE OF CONTENTS

1 GLOSSARY

Purpose: Eliminate ambiguity and keep wording consistent across technical, legal, and partner contexts.

Core terms	Definition
ABI	Application Binary Interface
AML	Anti-Money Laundry
Asset	A blockchain-native token or coin (e.g., stablecoin) controlled via a wallet's private keys.
AQL	Acceptance Quality Limit
API	Application Programming Interface
Batch ID	A unique identifier assigned to a manufacturing batch of recovery notes; used for traceability.
Bearer instrument	A transfer mechanism where possession alone is sufficient to redeem value; **out of scope for v1**.
CAPA	Corrective And Preventive Actions
CFT	Counter Financing of Terrorism
Chain-of-custody	Recorded sequence of custody handoffs for serialized media across factory, logistics, warehouse, and distribution points.
Checksum	Recorded sequence of custody handoffs for serialized media across factory, logistics, warehouse, and distribution points.
Companion app	The wallet application (e.g., Cryptocash Wallet app) providing UI for creation, verification, and recovery workflows; scoped for v1.
CRC	Cyclic Redundancy Code
CSPRNG	Cryptographically Secure Pseudorandom Number Generator; used to generate high-entropy keys.
CVSS	Common Vulnerability Scoring System
DNS	Domain Name Service
DPA	Data Protecting Act
DPIA	Data Protection Impact Assessment (data processing)
DRGB	Deterministic Random Bit Generator
ECC (Error Correction Code):	Mechanism enabling QR recovery when modules are damaged (e.g., QR ECC level).
ECDSA	Elliptic Curve Digital Signature Algorithm
Evidence artifact	A specific document or dataset (audit report, lab report, protocol) that substantiates a claim.
EVM	Ethereum Virtual Machine
FPGA	Field Programmable Gate Array
HSM	Hardware Security Module

GDPR	General Data Protection Regulation
Go/No-Go gate	A decision checkpoint where deployment proceeds only if defined evidence and acceptance criteria are met.
ID	Identification Data
IDTA	International Data Transfer Agreement
IPC	Information Protection and Control
JTAG	Joint Test Action Group
Key non-retention	Requirement that private keys/secret material are not stored, logged, or transmitted beyond the workflow window.
KPI	Key Performance Indicator
KYC	Know Your Customer
MCU	Micro Processing Unit
MITM	Man In The Middle
NDA	Non-Disclosure Agreement
Note (Recovery Note):	The physical recovery medium containing encoded recovery material; **recovery instrument in v1**.
Note serial number	A unique identifier printed on an individual note, enabling traceability and quarantine/recall by range.
OS	Operating System
OTA	Over The Air
Owner wallet	The user's primary wallet account context that receives recovered assets.
Owner binding	Mechanism restricting recovery so funds can only be recovered to owner-bound destinations (address/whitelist/proof).
PGP	Pretty Good Privacy
PoC	Proof-of-Concept
PoS	Proof-of-Stake
Post-print verification gate	Mandatory step that validates QR decoding and fallback checksum before finalizing issuance.
QR	Quick Response
RACI	Responsible, Accountable, Consulted, Informed
Recovery wallet / note wallet	A wallet created for a specific recovery note; holds assets intended for offline recovery.
Recover / Convert	The process by which the assets in the recovery wallet are recovered / converted back to their digital form.
ROM	Read-Only Memory
RoPA	Records of Processing Activities (General Data Protection Regulation)
RMA	Return Merchandise Authorization process for defective physical media or device components.
RNG	Random Number Generation

RPC	Remote Procedure Call
SCC	Standard Contractual Clauses
SDK	Software Development Kit
SWD	Serial Wire Debug
Secure boot	Hardware/firmware process ensuring only authenticated firmware executes on the device.
Secure element	Tamper-resistant hardware component used to store secrets and perform cryptographic operations.
SOP	Standard Operation Procedure
Supply-chain attack	Compromise introduced during manufacturing, provisioning, shipping, or distribution.
Tamper-evident	Design that provides visible evidence of opening or manipulation under defined inspection methods.
TBD	To Be Defined
TFR / Travel Rule	EU transfer-of-funds requirements extended to crypto transfers (relevant to regulated service providers; not v1 itself).
TON	The Open Network
TUF	The Update Framework
UART	Universal Asynchronous Receiver-Transmitter
UI	User Interface
URL	Universal Resource Locator
UX	User Experience
v1 (Stablenotes v1):	SDK-first recovery layer with owner-bound recovery; non-transactional by design.
v2 (Stablenotes v2):	Future bearer/offline transfer program; explicitly outside scope of this whitepaper.
VDP	Vulnerability Disclosure Policy
Whitelist	A permitted set of owner-bound destination addresses for recovery (if implemented).
Wi-Fi	IEEE 802.11x Wireless Networking (Wireless Fidelity)

Glossary maintenance rule: If a term appears in a heading or a critical control, it must be defined here.

2 PROBLEM STATEMENT & DESIGN RATIONALE

2.1 Problem Statement: Stablecoin adoption is real, but custody and resilience are brittle

Stablecoins and crypto assets have clear advantages over legacy rails: global reach, programmable settlement, and (in the case of stablecoins) a unit of account that can be more stable than local currency. In many emerging markets, stablecoins increasingly function as "digital dollars" used for savings, budgeting, and cross-border value transfer. However, adoption is constrained by a set of practical barriers that are not primarily technological—they are **operational, behavioral, and security-related**:

2.1.1 Storage safety is the primary failure mode

For many users, the real question is not "how do I transact?" but "how do I **not lose access**?" Funds are lost due to:

- lost or stolen phones,
- device failure,
- malware and phishing,
- forgotten credentials,
- accidental deletion,
- unsafe backup habits.

In other words: users may succeed at acquiring stablecoins but fail at keeping access to them.

2.1.2 Intermittent connectivity makes recovery harder when it matters most

Emerging-market usage often coincides with:

- unstable internet connectivity,
- limited device quality,
- constrained power access,
- disaster conditions or disruptions (storms, outages),
- shared devices and unsafe environments.

These conditions reduce the reliability of cloud-based backups and increase the chance that recovery is needed precisely when online access is degraded.

2.1.3 Device compromise is not an edge case

Self-custody on smartphones expands the threat surface:

- malicious apps and OS compromise,
- SIM swap / account takeover,

- screen overlay attacks,
- clipboard interception,
- “trusted contact” social engineering,
- fake wallet apps.

Even well-designed wallet software cannot fully remove the reality that end-user devices are frequently compromised.

2.1.4 Custodial alternatives solve usability, but create different failure modes

Custodial solutions (exchanges, hosted wallets) can be easier to use, but introduce:

- counterparty risk (insolvency, hacks),
- account freezes,
- withdrawal limits,
- policy changes and deplatforming,
- jurisdictional and regulatory exposure outside the user’s control.

This leaves users with an uncomfortable choice:

- **self-custody** (more control, more personal responsibility, higher likelihood of loss through backup failure), or
- **custody** (less personal burden, higher counterparty and access risk).

2.1.5 Core pain statement

Self-custody is hard; backups fail; offline resilience is poor.
Stablenotes v1 is designed to reduce the probability of irreversible loss caused by recovery failures, without moving users back into custody.

2.2 Design Rationale: Why printed recovery media exists

The Stablenotes v1 approach is based on a simple observation: digital assets are only as durable as the recovery methods used to secure them. In many real-world environments, “backup” is the weakest link—not blockchain security.

2.2.1 Why not hardware-only (e.g., traditional hardware wallets)?

Hardware wallets can provide excellent security properties, but they do not fully solve emerging-market adoption constraints:

- cost and availability,
- usability complexity (seed phrase management, transaction signing UX),
- device loss and replacement friction,
- dependency on supply chain and secure setup,
- user reluctance to carry or trust specialized devices.

Additionally, many users still end up writing down seed phrases on paper—often poorly—reintroducing fragility.

2.2.2 Why not custodial recovery (hosted wallets/exchanges)?

Custodial recovery is convenient but shifts the risk:

- the user trades loss-of-key risk for counterparty/access risk,
- introduces policy and jurisdictional uncertainty,
- and undermines the self-custody promise that motivates stablecoin adoption for many users.

For a large portion of the market, **the reason stablecoins matter** is independence from local and institutional fragility. Custody reintroduces that fragility.

2.2.3 Why paper-like recovery media persists (and why Stablenotes v1 focuses here)

Paper-like recovery media exists because it is:

- **offline** (does not require ongoing connectivity),
- **portable** (can be stored and transported),
- **human-auditable** (a user can visually inspect and handle it),
- **simple** (does not require a second computing device to exist),
- and can be designed to be **verifiable at creation time** (print + scan checks).

In other words, printed recovery media offers a form of *tangibility* and *portability* that complements digital assets—when engineered carefully.

Stablenotes v1 focuses on printing not initially a "payment note," but a **durable recovery instrument** created through a controlled workflow with defined assurance gates:

- strong randomness for key material,
- strict non-retention,
- survivability/legibility testing,
- post-print verification,
- and tamper-evidence baseline controls.

2.2.4 Tradeoffs and limitations (explicit)

Printed recovery media is not magic; it moves risks rather than completely eliminating them. Key trade-offs include:

- **Theft risk:** a physical artifact can be stolen. Users must treat it as sensitive recovery material.
- **Physical deterioration:** ink/substrate damage, stains, abrasion, humidity, and UV can degrade readability.
- **Handheld printing risk:** the mobile ATM device is a handheld printing device that requires initial training and education.
- **User handling risk:** folding, smudging, improper storage, or opening tamper layers prematurely can compromise security or recoverability.

- **Operational risk:** printing the blank recovery notes is a manufacturing process; quality control and traceability matter.
- **No perfect guarantees:** even strong tamper evidence does not prove no one copied a secret; it provides defined signals under defined assumptions.

Stablenotes v1 explicitly embraces these trade-offs by making them testable and operationally managed rather than ignored.

2.3 Design Goals and Non-Goals (v1)

2.3.1 Design goals (what v1 is built to achieve)

Stablenotes v1 is designed around five core goals:

1. **Survivability**
   Printed recovery media should remain readable and usable under defined real-world handling and storage conditions (with measurable acceptance criteria).
2. **Recoverability**
   Recovery should work reliably via QR plus a human-readable fallback (with checksums/integrity markers), with clear user guidance and support processes.
3. **Non-retention**
   Secret material must not be retained by the app/SDK/device beyond the minimum workflow window. No leakage through logs, telemetry, crash dumps, or storage.
4. **Tamper-evidence (baseline)**
   If a tamper-evident layer is used, it must provide clear, testable "opened vs intact" signaling and be compatible with long-term legibility.
5. **Simplicity**
   The workflow must be understandable by non-expert users, with enforced verification steps and minimal opportunities for irreversible user error.

2.3.2 Non-goals (explicit exclusions to preserve scope and compliance posture)

Stablenotes v1 explicitly does **not** aim to be:

- **A transaction rail** (no payments network; no transfer protocol; no attempt to replace existing rails)
- **An anonymity tool** (no attempt to bypass AML/CFT regimes or conceal flows)
- **A cash replacement** (no "spend like cash" positioning)
- **A mass distribution program** (no uncontrolled large-scale physical circulation model)
- **An operator subsidy model** (no operator network, no "cash-out" agent rollouts)
- **A token sale or tokenomics vehicle** (no fundraising token; no revenue share token; no speculative issuance)

These non-goals are not philosophical—they are practical risk controls to keep v1 deployable as an SDK-first recovery layer.

Design Principles (Stablenotes v1)

These principles govern Stablenotes v1 design decisions and serve as the default tie-breakers when requirements compete.

1. **Evidence over claims**
No material security or reliability claim is "true" until it is backed by an audit, lab report, or repeatable test protocol.

2. **Boundaries over narratives**
v1 is a recovery layer. It must be technically and operationally hard to repurpose as a payments rail, regardless of marketing.

3. **Non-custodial by construction**
The system must be architected so PrintDreams cannot access user private keys or funds and cannot unilaterally move assets.

4. **No irreversible step without verification**
Any workflow that could lead to irreversible loss (e.g., finalizing a recovery note) must include a mandatory verification gate (QR scan + fallback integrity check).

5. **Minimize secret lifetime**
Secret material should exist only as long as required to complete the workflow, with strict non-retention and memory hygiene controls.

6. **Assume the phone is hostile**
Smartphones are frequently compromised. Design should reduce exposure to clipboard, logging, backgrounding, and third-party SDK leakage.

7. **Recoverability beats elegance**
A recovery system that fails rarely is better than a cleaner design that fails occasionally. Redundancy and checksums are features, not clutter.

8. **Physical durability is a security property**
Legibility and survivability are part of security. Unreadable keys are equivalent to stolen keys from the user's perspective.

9. **Controlled supply chain, controlled blast radius**
Serialized media, chain-of-custody logging, and quarantine-by-serial-range must allow fast containment of defects or compromise.

10. **Safe defaults, explicit escalation**
Default settings should prioritize safety and clarity. Any advanced behavior must be user-initiated, explicit, and well-disclosed.

11. **Partner integration must be certifiable**
SDK integrations should have a "golden path," a certification checklist, and clear responsibility boundaries to prevent partner-side regressions.

12. **Design for failure modes, not ideal use**
The system should be evaluated against realistic failure modes (damage, loss, scams, user error, interruptions) rather than ideal lab conditions.

3 SYSTEM OVERVIEW -ARCHITECTURE AND TRUST BOUNDARIES

3.1 Components

Stablenotes v1 comprises five main component groups:

3.1.1 Wallet application and SDK Cryptocash Wallet App (reference implementation)

A self-custodial wallet application that:

- securely stores a “first private key” for a source wallet (e.g., in a secure element),
- initiates “Create Recovery Note” workflows,
- creates a “second private key” for a new wallet,
- derives the corresponding public key/address,
- signs a transaction moving a digital asset into that new wallet, and
- interacts with the printing device, and
- signs/broadcasts blockchain transactions as needed (depending on integration mode).

Stablenotes SDK (partner integration)
A software toolkit that enables third-party wallets to implement:

- note creation orchestration,
- payload encoding (QR + fallback text),
- verification gates,
- note lifecycle state tracking (optional),
- and recovery/import flows.

3.1.2 Printing device (“printer unit”) and secure execution environment

A dedicated mobile printing device branded as “Cryptocash ATM” (or a host device controlling a print module) that performs the core functions described in the patent reference:

- prints the second private key (and associated public data) onto the physical note, then deletes any digital copy of the second private key after confirming that the printing process has been successfully performed.

The main patent of the Stablenotes project (WO2025237814A1) explicitly describes using a **secure element** for secret storage and provides an example secure element component. The secure element will be implemented for v2 and it is not available for v1.

3.1.3 Physical recovery media (“note”)

An originally semi-blank physical note that by using the mobile printing device prints the following artifacts (at minimum):

- a machine-readable **public identifier** (public key/address) for monitoring or verification, and
- a protected recovery secret (e.g., private key or recovery material) in **QR and/or human-readable** form.

The patent describes printing both the second public key and second private key on the same sheet and optionally adding:

- asset identifier,
- timestamp,
- and other metadata.

Serialization and traceability (v1 operational control)
Notes are assumed to be uniquely serialized and batch-serialized for traceability. The patent reference describes associating keys to a unique serial number pre-printed on the physical note as a condition of signing the transaction.

This is to ensure that the private keys are not printed on a random piece of paper but on a genuine recovery note and security sticker engineered to last long and endure the toughest environmental conditions.

3.1.4 Tamper-evident security label (optional in v1; baseline claims only)

A two-layer tamper-evident sticker with an inkjet-friendly printing substrate may have a secret area to:

- obscure readability before opening, and
- show irreversible “void” evidence upon opening.

The patent describes a security sticker with an obfuscation area and visible residues indicating the note has been opened [1]. Implementation in v1 has two purposes:

1. Clear indication to the user that the funds in the note have already been recovered / converted into their original digital format
2. Prepare deployment of v2 by field testing of this feature

3.1.5 Blockchain network and smart contracts

Stablenotes v1 is chain-agnostic by design but is expected to support deployments where assets are held as:

- native chain assets and/or
- fungible token standards.

TON (initial chain)
TON uses a Proof-of-Stake validator model.

[1] *v1 scope note: this supports “tamper-evident baseline” claims only; it is not a v1 claim that copying is impossible.*

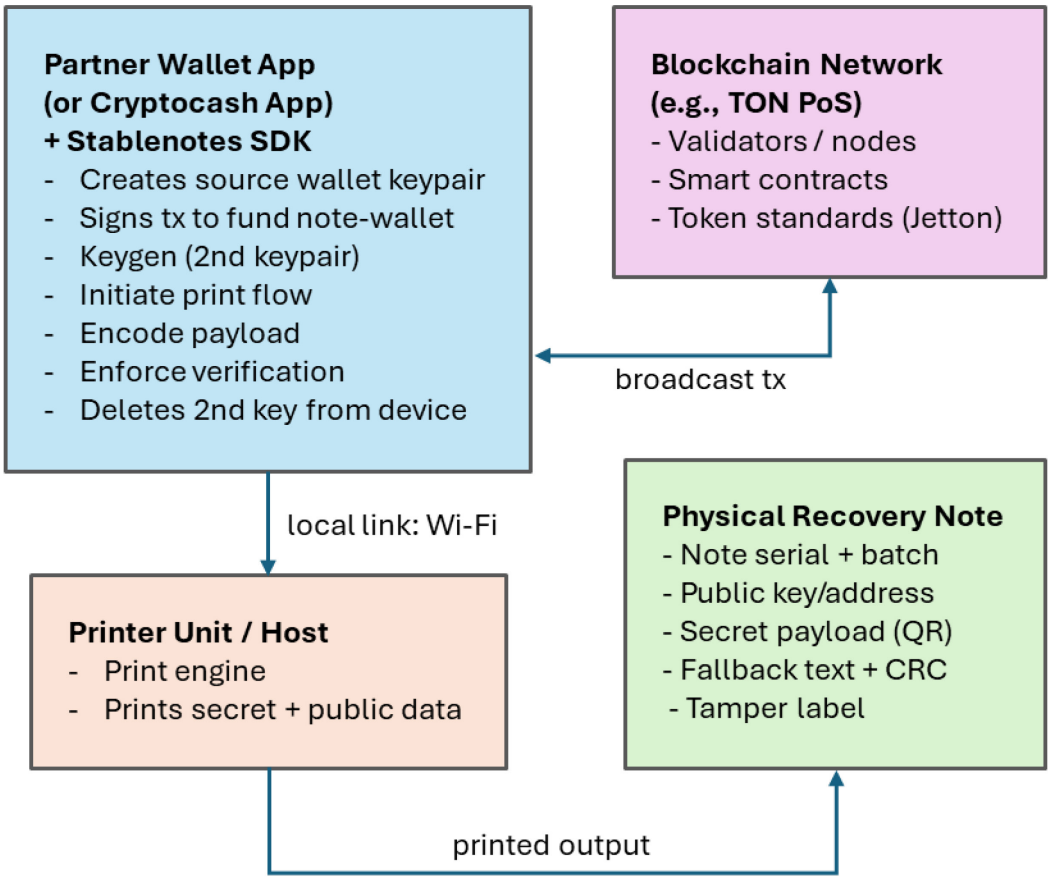
Stablecoins on TON are commonly implemented as **Jettons** (TEP-74), TON's analogue to ERC-20.

3.2 High-level workflow (v1 recovery use case)

Stablenotes v1 uses the "print-to-new-wallet" pattern described in the patent, but positions it strictly as **cold storage / recovery**, not bearer transfer. At a high level:

1. User initiates a "Create Recovery Note" workflow in the wallet app (or partner app via SDK).
2. Printing device (or its host) creates a new keypair (second private key + derived public key/address).
3. A transaction is signed to move a specified amount of the asset from a source wallet into the newly created wallet associated with the printed note.
4. The device prints the recovery payload onto the note and then deletes the digital copy of the second private key.
5. The user stores the note offline as recovery media.

3.3 Architecture diagram (logical view)



3.4 Trust model: boundaries and assumptions

Stablenotes v1 is designed with explicit trust boundaries. The goal is to make it easy to reason about **what must be trusted** and what is treated as hostile.

3.4.1 Trusted components (must be trusted for security claims to hold)

A) Printer unit secure execution path

- Secure element (or equivalent secure storage) used for any persistent secret storage (not present in v1).
- Firmware integrity (secure boot, signed updates)
- Correct key generation and deletion behavior as implemented and verified

B) Printing and media production

- Printing process quality gates (print fidelity and verification at creation time)
- Media specification (ink/substrate) meets survivability requirements
- Serialization integrity (unique serials; batch manifests; no duplicates)
- Tamper label materials behave as specified (obfuscation + irreversible open evidence).

3.4.2 Untrusted components (assumed potentially hostile)

A) Smartphone environment

The phone is treated as **untrusted** by default:

- malware, compromised OS, malicious keyboards, clipboard scraping,
- risky third-party SDKs (analytics/session replay),
- phishing and UI overlay attacks.

v1 therefore assumes:

- key material must not be logged or persisted unintentionally (non-retention/memory hygiene),
- the app must enforce verification gates and limit exposure during sensitive flows.

B) Public networks and external infrastructure

- Wi-Fi/mobile networks are untrusted (MITM, DNS manipulation).
- Public blockchain nodes and RPC endpoints may be malicious or unreliable.
- Token spoofing risks exist (e.g., counterfeit jettons on TON if the wallet does not verify master addresses).

3.4.3 Out of scope (explicit)

The following are outside the v1 security scope unless otherwise stated in a written assurance addendum:

- attacks on underlying cryptographic primitives (e.g., ECDSA break),
- catastrophic compromise of the underlying blockchain consensus (e.g., chain halt / finality failure),
- coercion/extortion of end users (physical threats),
- nation-state level supply-chain compromise unless explicitly tested and mitigated,
- guarantees of anonymity or untraceability (v1 is not an anonymity system),
- v2 “bearer transfer” adversarial transfer resistance (explicitly excluded from v1).

3.5 Threat model (high-level summary)

A full threat model is provided in the Security & Assurance chapter and appendices. At a high level, v1 must be resilient against:

1. **Malware on the phone**
- attempts to exfiltrate secrets via logs/telemetry/clipboard,
- UI deception during print or recovery,
- transaction manipulation.
2. **Physical theft**
- theft of printed recovery notes,
- theft of the printer unit,
- theft during shipment/distribution (media stock).
3. **Insider threats**
- insider at factory or warehouse substituting media or leaking serial ranges,
- insider at partner retail or distributor diverting stock,
- insider at software vendor introducing telemetry leakage.
4. **Supply chain compromise**
- counterfeit media injection,
- tamper-seal bypass or repacking,
- compromised device firmware or provisioning process.
5. **Counterfeit token / token identity spoofing**
- relevant on TON, where look-alike jettons can exist; wallet must verify authenticity by master address/allowlist.

3.6 Data Flows and Privacy Boundary

This subsection defines what data exists in Stablenotes v1, where it flows, and what must never leave the user’s control. It is written for partner compliance/security review and to prevent accidental “creep” into custodial or regulated-service territory.

3.6.1 Purpose and scope

This section documents the **data flow model** and the **privacy boundary** for Stablenotes v1 in an SDK-first deployment.

It covers:

- key material and recovery payload handling,
- device and app telemetry boundaries,
- partner integrations (SDK embedding),
- third-party service integrations (e.g., on/off-ramps) insofar as they interact with v1.

It does **not** replace the Privacy Notice, DPA, or GDPR documentation, which are provided separately.

3.6.2 Data classification (summary)

Stablenotes v1 uses a strict classification model consistent with the Security & Assurance chapter:

Class A — Critical secrets (must never leave trusted execution boundary)

- private keys / seed material
- raw entropy
- unencrypted recovery payload prior to print
- any derived secret material used to reconstruct a private key

Class B — Sensitive but not secret

- public addresses/public keys
- note serial numbers and batch IDs (context-dependent)
- device identifiers (if used for integrity checks)
- transaction hashes and chain metadata

Class C — Operational / non-sensitive

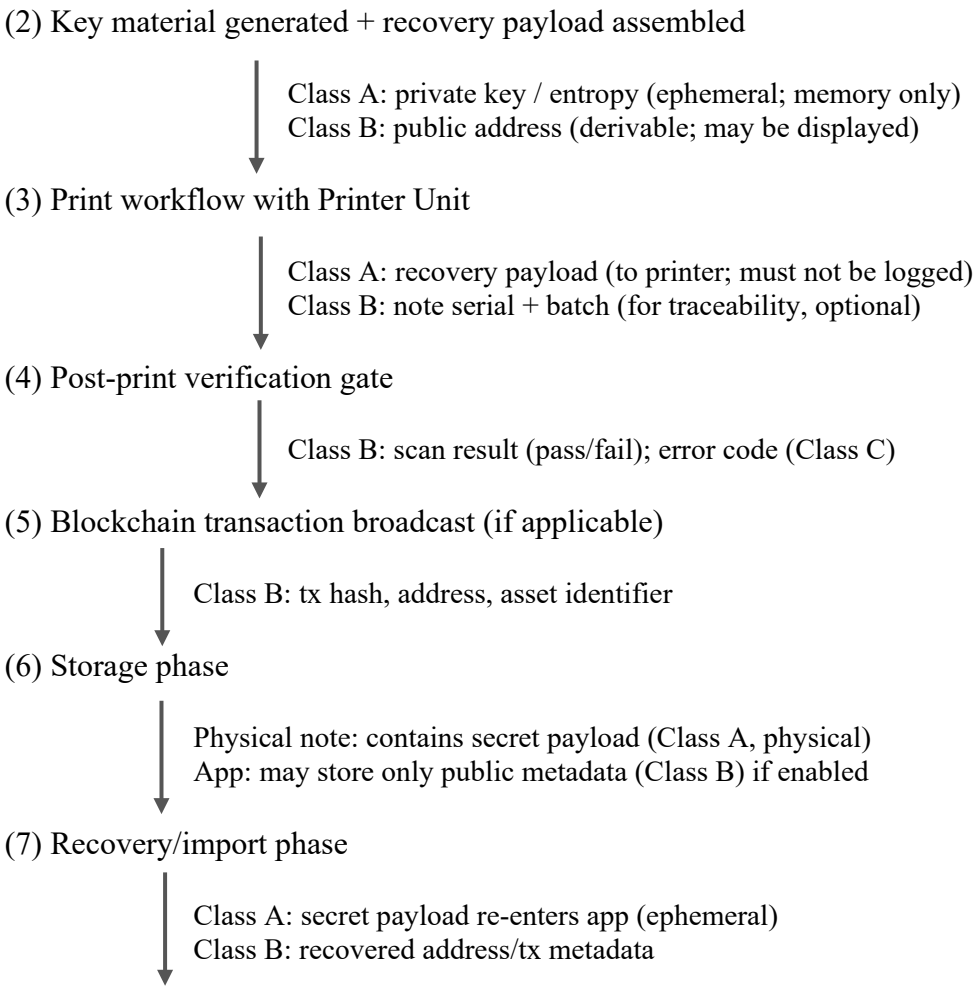
- feature usage events (non-sensitive)
- performance metrics
- generic error codes without sensitive context

Rule: Only Class C may be collected as telemetry by default. Any Class B collection must be strictly minimized and justified. Class A is prohibited from collection, storage, or transmission.

3.6.3 High-level data flow map

(1) User initiates "Create Recovery Note" in Wallet App

Class C: UI events (optional, local)



3.7 Privacy boundary (non-negotiable)

3.7.1 What PrintDreams must never receive or store

PrintDreams must not receive, transmit, or store:

- private keys, seed phrases, or any Class A secret material
- recovery payload contents (even encrypted, unless the encryption is exclusively user-controlled and PrintDreams never sees keys)
- KYC/AML identity documents or biometric data (if third-party ramp is embedded)

If PrintDreams servers ever receive Class A data—even accidentally via crash dumps, logs, or analytics—then the "software not a bank" positioning becomes legally fragile and the security posture becomes indefensible.

3.7.2 What may be stored locally on-device (allowed, minimal)

Allowed local storage (app side) should be limited to:

- public addresses and display labels (Class B)
- transaction hashes and chain state (Class B)
- device/app configuration (Class C)
- user consent logs (Class C) *(e.g., acceptance of disclosures; no sensitive content)*

3.7.3 What may be transmitted off-device (default should be "nothing sensitive")

Permissible outbound transmission should be constrained to:

- Class C operational telemetry (opt-in where possible)
- minimal Class B where required for anti-fraud or support, and only with explicit disclosure and governance

Explicit prohibitions:

- no session replay on sensitive flows
- no screenshot capture of recovery payload displays
- no debug logging in production

3.9 Third-party service boundary (e.g., MoonPay on/off-ramp)

Stablenotes v1 may be demonstrated alongside buy/sell features provided by regulated third parties. In that model:

3.9.1 Controller separation (high-level)

- Third-party ramp provider acts as the controller for KYC and transaction processing data
- Cryptocash/PrintDreams should receive only:
    - tokenized status callbacks (success/failure)
    - asset/amount confirmation necessary to update wallet UI
    - no raw KYC artifacts

3.9.2 Allowed data exchange (example)

Allowed:

- wallet address (public) for delivery of purchased assets
- transaction status + reference ID
- country eligibility flags

Not allowed:

- identity document images
- biometric liveness data
- raw sanctions screening results beyond high-level pass/fail flags

- payment card data (unless handled entirely by third-party's PCI environment)

3.10 Note serialization and traceability data

If Stablenotes v1 implements serialized media tracking:

- Note serials and batch IDs are **Class B** data.
- They may be stored and transmitted for supply-chain integrity purposes, but should be:
  - minimized (avoid linking serials to personal identities unless necessary)
  - protected with access controls
  - governed by retention limits

Design principle: Traceability should focus on **inventory integrity**, not user surveillance.

3.11 Telemetry and diagnostics (recommended governance)

A v1 deployment should implement:

3.11.1 Telemetry allowlist

Only allowlist fields may be collected, such as:

- SDK version
- device model (coarse)
- pass/fail codes for verification gate
- generic error codes
- performance timing metrics

3.11.2 Prohibited telemetry

Prohibit collection of:

- any QR contents
- any secret material
- full addresses unless explicitly required (prefer hashed/pseudonymized where possible)
- screen recordings/session replay in key-handling screens

3.11.3 Crash reporting controls

- disable memory dumps where possible
- scrub sensitive contexts
- avoid attaching app state snapshots that may include key material

---

3.12 Out of scope (explicit)

The privacy boundary in this section does not attempt to address:

- law enforcement requests handling (covered in separate policy)
- user-driven sharing of data (e.g., user screenshots)
- advanced analytics for fraud scoring using AI (requires separate legal review and may trigger EU AI Act obligations)

3.13 Implementation placeholders (must be resolved before partner deployment)

- **Signing model:** App signs tx for v1, Device signs tx for v2
- **Connectivity:** Specifics for device communication [Wi-Fi for v1, BLE/Wi-Fi for v2]
- **Telemetry stack:** [TBD], including third-party SDK inventory and session replay status
- **Ramp integration mode:** [Redirect / Embedded / Native SDK] and exact data boundaries
- **Supply chain system-of-record:** [TBD] event log schema and retention
- **Privacy documentation:** controller matrix, RoPA excerpt, DPIA triggers, SCC/IDTA needs
- **Supported chains:** [TON / Ethereum / Others TBD]
- **Supported stablecoins/assets:** [USD₮ (Jetton) / Others TBD] with allowlist policy
- **Owner-binding mechanism (if adopted):** [TBD]
- **Device security controls:** [secure boot, signed updates, audit references TBD]
- **Supply chain assurance artifacts:** [serialization spec, manifests, chain-of-custody SOPs]

4 ASSET MODEL & WALLET LIFECYCLE (V1 OWNER-BOUND RECOVERY)

This chapter defines how Stablenotes v1 represents assets across two wallet contexts and enforces a recovery-first model. The goal is to make v1 **non-transactional by design**, so that the printed recovery instrument functions as a **backup and recovery tool**, not a bearer payment instrument.

4.1 Wallet model

4.1.1 Owner wallet

The **Owner Wallet** is the user's primary self-custodial wallet (or account context) within a partner wallet application. It is the wallet the user normally uses to hold and manage crypto assets and stablecoins.

Characteristics:

- Controlled exclusively by the user's private keys or recovery method chosen by the user.
- Holds the user's working balances.
- Is the destination for any recovery operation performed using Stablenotes v1.

4.1.2 Recovery wallet / note wallet

The **Recovery Wallet** (also referred to as the **Note Wallet**) is a dedicated wallet created specifically to hold assets intended for offline recovery. In Stablenotes v1, it is represented physically by the printed recovery instrument ("note"), which encodes the secret material required to recover those assets.

Characteristics:

- Created during the "Create Recovery Note" workflow.
- Receives a defined amount of an asset (e.g., a stablecoin) from the Owner Wallet or other permitted source.
- Is intended to be **stored offline** as part of a resilience strategy.
- Is subject to **owner-bound recovery constraints** (Section 4.2).

4.1.3 Explicit v1 limitation (non-transactional intent)

Stablenotes v1 notes are not intended for P2P bearer transfer.
They are **recovery instruments**. The v1 specification is designed to prevent or materially impede use as a transferable cash substitute, regardless of user intent or marketing.
Any "bearer transfer" feature set is outside v1 scope.

4.2 Owner binding mechanism (non-transactional by design)

4.2.1 Objective

The owner binding mechanism ensures that a v1 note cannot be used as a generic bearer instrument where "whoever holds it can redeem it freely." Instead, recovery is restricted to the legitimate owner context.

4.2.2 Approved binding patterns (choose one for v1)

Stablenotes v1 supports the following binding patterns. The actual choice MUST be made explicit per deployment.

Pattern A — Pre-committed owner address (recommended default)

At note creation time, the Recovery Wallet is bound to a **pre-committed Owner Address** (or set of addresses). Recovery operations are only permitted to send assets to the committed Owner Address.

- The binding data is created and recorded during note issuance.
- The recovery workflow enforces the destination constraint (e.g., via smart contract logic, signed policy, or wallet enforcement rules depending on architecture).

Pros: simplest, strongest "non-bearer" posture.
Cons: if the owner address changes, migration requires explicit re-issuance.

Pattern B — Whitelist of owner addresses (controlled set)

A note is recoverable only to a whitelist of addresses controlled by the owner.

- Suitable when users have multiple devices or a staged migration plan.
- Changes to whitelist are allowed only through a protected process (e.g., by using the Owner Wallet and additional confirmation).

Pros: flexibility for real users.
Cons: additional complexity and governance.

Pattern C — Proof-of-ownership requirement (cryptographic proof)

Recovery requires the user to prove ownership of the Owner Wallet (e.g., by providing a signature from the Owner Wallet keys) before the recovery transaction can execute.

Pros: strong conceptual model; supports owner address rotation.
Cons: depends on reliable signature availability and UX; can be brittle in emergencies.

4.2.3 Why owner binding prevents bearer-payment use (even if users try)

A bearer instrument works because possession alone enables redemption or spending. Owner binding breaks that property:

- Possession of the note is **insufficient** to move funds arbitrarily.
- A third party cannot "cash out" or spend the note's contents unless they can also satisfy the owner-bound constraint (pre-committed address, whitelist membership, or proof-of-ownership).

- Even if a user physically hands the note to someone else, the recipient cannot redeem the funds to their own address unless the owner explicitly intended and configured that outcome.

This turns the physical note into a **recovery tool** rather than a **transferable payment object**.

4.2.4 Residual risks (not eliminated by binding)

Owner binding reduces bearer use, but does not eliminate:

- theft + coercion (forcing owner to cooperate),
- user mistakes in setting the owner binding,
- compromise of the owner wallet itself.

These risks are handled through user guidance, secure defaults, and partner wallet security controls.

4.3 Lifecycle states (Created → Funded → Stored → Recovered → Retired)

Stablenotes v1 defines a simple lifecycle. Each state has **allowed actions** and **forbidden actions** to preserve the recovery-only posture.

4.3.1 State 1 — Created

Definition: A Recovery Wallet is created and bound to the owner (Section 4.2) but not yet funded.

Allowed actions:

- Generate Recovery Wallet key material (per RNG + non-retention requirements).
- Encode recovery payload for printing.
- Assign note serial/batch metadata (optional).
- Print the recovery instrument.
- Perform mandatory post-print verification (scan + checksum validation).

Forbidden actions:

- Broadcasting any funding transaction before verification passes.
- Persisting any Class A secret material beyond the workflow window.
- Creating a note without owner binding.

4.3.2 State 2 — Funded

Definition: Assets have been transferred into the Recovery Wallet.

Allowed actions:

- Display balance and public address for monitoring (optional).

- Confirm funding transaction finality and record tx hash.
- User may choose to store the note offline.

Forbidden actions:

- Arbitrary transfers to third-party addresses (unless owner-binding explicitly permits, which v1 should not).
- Integration of "spend flows" or merchant redemption UX.
- Any automated rebalancing or yield activity.

4.3.3 State 3 — Stored

Definition: The recovery instrument is stored offline for resilience purposes.

Allowed actions:

- Periodic optional verification (e.g., scan-check without exposing secrets, if supported).
- Safe storage guidance and replacement planning.

Forbidden actions:

- Treating the note as a payment object in marketing, UI, or partner distribution.
- Re-opening and re-storing an opened note as if it remains secure (if a tamper layer is used).
- Any "hot wallet convenience" behavior that increases secret exposure.

4.3.4 State 4 — Recovered

Definition: The note has been used to restore access and transfer funds to the owner-bound destination.

Allowed actions:

- Import/scan recovery payload in controlled recovery flow.
- Transfer assets from Recovery Wallet to the Owner Wallet destination as permitted by binding.
- Record recovery event for user history (without exposing secrets).

Forbidden actions:

- Recovery to an unbound address.
- Partial recovery behavior that leaves the Recovery Wallet active without clear user intent (unless explicitly supported and disclosed).

4.3.5 State 5 — Retired

Definition: Recovery is complete and the Recovery Wallet is no longer intended for storage.

Allowed actions:

- Mark note as retired in the app.
- Provide user instructions for safe disposal or archiving.

- Optional: update supply-chain status (e.g., “issued → retired”) if serial tracking is enabled.

Forbidden actions:

- Reusing a retired note for new storage without a full re-issuance workflow.
- Storing or circulating a retired note as if it still protects value.

Implementation placeholders (must be resolved per deployment)

- Selected owner-binding pattern: **[A/B/C]**
- Enforcement layer: **[Smart contract enforcement / wallet enforcement / hybrid]**
- Recovery semantics: **[sweep-all / partial recovery]**
- Whether note lifecycle state is tracked on-chain, in-app, or off-chain: **[TBD]**
- Whether serial/batch traceability is linked to user identity (recommended: **no**, unless required): **[TBD]**

5 SMART CONTRACT SPECIFICATION (STABLENOTES V1)

This chapter defines the on-chain components used by Stablenotes v1. Because partner deployments may vary by chain and custody model, this chapter is written as a reference specification with explicit placeholders. Any production deployment MUST populate the placeholders and publish a chain-specific annex.

5.1 Contracts and interfaces

5.1.1 Supported chains and contract addresses

Stablenotes v1 is designed to support one or more chains where assets can be held and moved via smart contracts or standard token contracts.

Chain support (initial):

- **TON Network: [Supported / TBD]**

Contract addresses (per chain):

- TON:
  - RecoveryPolicyContract (if used): **TBD at deployment**
  - NoteRegistryContract (optional): **TBD at deployment**
  - Allowlist/TokenVerifier (optional): **TBD at deployment**
- Additional chains (if any): **TBD**

5.1.2 Contract roles and responsibilities (logical)

Depending on the chosen owner-binding enforcement, Stablenotes v1 may include:

1. **Recovery Policy Contract (recommended)**
   Enforces that any “recovery” transfer from a note/recovery wallet can only be executed to an owner-bound destination (pre-committed address/whitelist/proof).
2. **Note Registry Contract (optional)**
   Stores minimal metadata about notes (e.g., note ID hash, creation timestamp, state markers).
3. **Token Allowlist / Verifier (optional; chain-specific)**
   Helps ensure the wallet interacts only with verified token contracts (e.g., verified stablecoin master contracts).

5.1.3 ABI / interface summaries (placeholders)

Below are interface summaries in a chain-agnostic style. For TON deployments, these must be mapped into TON contract message handlers and payload formats.

A) RecoveryPolicyContract (if used)

State variables (conceptual):

- owner_binding: one of:
  - owner_address (Pattern A)
  - whitelist (Pattern B)
  - proof_policy (Pattern C)
- status: Active / Recovered / Retired (optional)
- policy_version (if upgradeable)

Functions / messages (conceptual):

- bind_owner(note_wallet, owner_binding_params)
  Binds the recovery wallet to the owner destination policy.
  Called at creation time; must be immutable or tightly controlled after binding.
- recover(note_wallet, to_owner_address, amount, proof)
  Executes recovery transfer from note wallet to owner destination.
  MUST enforce owner binding.
- set_whitelist(note_wallet, whitelist_addrs, proof) *(Pattern B only)*
  Updates whitelist under strict authorization rules.
- retire(note_wallet, proof)
  Marks note wallet as retired (optional state tracking).

Events (conceptual):

- OwnerBound(note_wallet, binding_type, binding_hash, timestamp)
- Recovered(note_wallet, to, amount, tx_ref, timestamp)
- Retired(note_wallet, timestamp)
- WhitelistUpdated(note_wallet, whitelist_hash, timestamp) *(if applicable)*

Privacy note: events must not emit secret material. Binding parameters should be represented as hashes where practical.

B) NoteRegistryContract (optional)

Functions / messages:

- register_note(note_id_hash, note_wallet, metadata_hash)
- set_status(note_id_hash, status, proof)
- get_note(note_id_hash) (view/query)

Events:

- NoteRegistered(note_id_hash, note_wallet, timestamp)
- NoteStatusChanged(note_id_hash, status, timestamp)

C) Token allowlist / verifier (optional)

Functions / messages:

- is_verified_token(token_contract_address) -> bool
- verify_token(token_contract_address, proof) *(governance controlled)*

Events:

- TokenVerified(token_contract_address, timestamp)
- TokenRevoked(token_contract_address, timestamp)

5.2 Security properties

5.2.1 Core invariants (must hold in every deployment)

Invariant 1 — Recovery only to owner-bound destinations

Recovery of assets from the note/recovery wallet MUST be restricted to the owner-bound destination defined at creation time, using one of:

- pre-committed owner address,
- whitelist,
- or proof-of-ownership policy.

Requirement: Any transaction that attempts recovery to an unbound destination MUST fail deterministically.

Invariant 2 — No hidden admin drain routes

There MUST be no privileged function that allows PrintDreams, a partner, or any administrator to:

- move user assets from note wallets,
- override owner binding,
- or redirect funds without the owner satisfying the recovery policy.

Requirement: If any privileged keys exist (e.g., for upgrades), they must not be capable of moving user funds or altering recovery destinations in a way that enables confiscation or redirection.

Invariant 3 — Explicit upgradeability policy

The deployment MUST declare one of the following policies:

Policy A: Non-upgradeable (preferred for trust-critical contracts)

- Contracts are immutable; fixes require deploying new contracts and migrating via explicit user action.

Policy B: Upgradeable with strong controls
If upgradeability is required (e.g., chain constraints or partner requirements), the system MUST define:

- who controls upgrades (multi-sig, governance),
- timelocks,
- public announcement requirements,
- audit requirements for new versions,
- and a rollback/disable mechanism.

Requirement: Upgrades must not introduce a pathway to break Invariant 1 or Invariant 2.

5.2.2 Administrative roles (if any)

If any roles exist (strongly discouraged unless necessary), they MUST be defined explicitly:

- UpgradeAdmin (multi-sig only; timelocked)
- TokenVerifierAdmin (if allowlist exists; multi-sig)
- EmergencyPause *(optional; see below)*

Emergency pause (optional):
An emergency pause may be allowed only if:

- it cannot seize funds,
- it is narrowly scoped (e.g., pause new note registrations, not recovery),
- it has clear criteria and expiration rules,
- and it is disclosed to partners/users.

5.2.3 Audit and verification requirements

For any deployment using custom contracts:

- at least one independent security review MUST be completed pre-production,
- critical/high issues MUST be remediated and re-tested,
- and the audit report (or executive summary) MUST be available to integration partners under NDA.

5.3 Failure modes

This section describes expected behavior under adverse conditions. A deployment must document exact outcomes for each chain and integration model.

5.3.1 Chain congestion / high fees / delayed finality

Risk: Transactions may be delayed, fail, or become expensive.

Expected behavior:

- Recovery attempts may remain pending until included in a block.
- Wallet UX MUST show:
  - pending state,
  - estimated fee impact,
  - and a safe retry workflow that avoids duplicate conflicting transactions.
- If recovery is “sweep-all,” the UX MUST prevent repeated attempts that could create partial state confusion.

Mitigations:

- Use fee estimation with conservative buffers.
- Provide deterministic idempotency in contract logic (e.g., once recovered, subsequent recover calls fail or no-op).
- Provide user guidance for network congestion scenarios.

5.3.2 Partial transactions / interrupted workflows

Risk: The workflow may be interrupted after some steps succeed:

- note created but not funded,
- funded but not printed,
- printed but verification not recorded,
- recovery initiated but not completed.

Expected behavior:

- The system MUST define safe state transitions and rollback guidance:
  - If not funded: note is void; do not store as meaningful recovery.
  - If funded but print failed: funds remain in recovery wallet; recovery path depends on whether the key exists and is recoverable.
  - If printed but verification failed: treat as unsafe; re-issue recommended.
- Contracts (if used) SHOULD ensure state is consistent and queryable.

Mitigations:

- Enforce “no funding before verification” where possible.
- Store minimal state marker in contract/registry to detect incomplete issuance.
- Provide a support-assisted recovery procedure that never requires revealing secrets to PrintDreams.

5.3.3 Lost note (user cannot access recovery media)

Risk: User loses the physical note.

Expected behavior:

- If the Recovery Wallet is truly controlled only by the printed secret material, loss of the note means:
  - recovery is impossible, and funds in the Recovery Wallet may be permanently inaccessible.
- The product MUST disclose this clearly.

Mitigations (non-custodial only):

- Encourage users to create redundant recovery instruments if appropriate (policy-defined).
- Allow multiple owner-bound notes (separate wallets) rather than single single-point-of-failure.

- Optional: implement “time-locked fallback recovery” only if it does not introduce custody or admin drain risk *(generally discouraged for v1 unless carefully designed and audited).*

5.3.4 Compromised owner wallet

Risk: If the Owner Wallet is compromised, an attacker may satisfy owner-binding controls (e.g., sign proofs) or receive recovered funds.

Expected behavior:

- Owner-binding does not protect against compromise of the owner itself.
- If the owner wallet is compromised, recovery may send funds to an attacker-controlled address.

Mitigations:

- Partner wallets should support stronger owner security (PIN/biometrics, secure enclave signing where available, device attestation).
- Provide user guidance: if owner wallet compromise is suspected, migrate funds and re-issue recovery notes immediately.
- Consider allowing whitelist binding to multiple owner addresses where users maintain a “backup owner” device *(with strict UX and education).*

5.4 Deployment placeholders (must be completed for any production release)

- Contract deployment addresses: **[TBD per chain]**
- Final ABI/message formats (TON): **[TBD]**
- Owner binding enforcement method: **[on-chain / wallet-enforced / hybrid]**
- Upgradeability policy: **[Immutable / Timelocked upgradeable]**
- Emergency controls (if any): **[None / Defined scope]**
- Audit references: **[Auditor, date, report ID]**

6 SECURITY AND ASSURANCE

Stablenotes is a security product before it is a convenience product.
The core promise of Stablenotes v1 is narrow and testable: a user can generate and store a recovery instrument for tokenized assets and later recover those assets reliably. That promise fails if any of the following fail:

- the private key is generated with weak randomness,
- the key is retained or leaked by software,
- the printed recovery media degrades and becomes unreadable,
- tamper evidence is overstated or ambiguous,
- or a partner integration introduces unsafe behavior around the SDK.

This chapter defines the controls, boundaries, and evidence required to make that promise credible.

It is intentionally written as a set of requirements and ship gates, not aspirational claims. The objective is to make Stablenotes v1 deployable in an SDK-first model where partners can integrate the recovery layer without inheriting unknown security or liability risk.

Scope and philosophy

This chapter covers six assurance domains:

1. **RNG and key generation assurance** — keys must be unpredictable and uniquely generated.
2. **Key non-retention and memory hygiene** — secrets must not persist in logs, storage, telemetry, or crash paths.
3. **Print survivability and key legibility** — the recovery instrument must remain recoverable under defined real-world conditions.
4. **Tamper evidence and adversarial access** — v1 tamper claims must be specific, limited, and testable.
5. **Serialized Note Supply Chain and Distribution Assurance** —a traceable, tamper-evident, low-complexity supply chain for Stablenotes note media from factory → warehouse → distribution points → retail issuance.
6. **Partner integration security boundary and liability model** — security and operational responsibility must follow control.

The chapter also distinguishes between:

- **v1 (Cold Storage / Recovery)**: owner-controlled recovery media, with limited and evidence-backed claims.
- **v2 (Bearer / Transfer)**: a future program requiring separate adversarial testing, additional controls, and separate go/no-go gates.

This separation is deliberate. It prevents v1 from carrying security or regulatory claims that belong to a later, more demanding product phase.

Evidence-first standard

Every material security or reliability claim in this chapter is expected to map to an artifact in a Claims-to-Evidence matrix (audit report, lab report, protocol, or test record). A claim without a corresponding artifact is not considered production-ready.

This is not just a technical preference. It is an operating requirement for partner integrations, investor diligence, incident response, and future certification or regulatory review.

What this chapter is not

This chapter does not define smart contract business logic or token economics, jurisdiction-specific regulatory strategy or commercial go-to-market tactics.

Those topics are covered in other chapters. Here, the focus is narrower: **what must be true, and what must be proven, for Stablenotes v1 to be trusted**.

Deployment implication for Option A

Under the SDK-first strategy, Stablenotes scales through partner integrations rather than direct consumer distribution. That increases the importance of clear boundaries and repeatable assurance:

- partners need a defined integration path,
- support teams need predictable failure handling,
- and liability must be allocated based on which party controls the failing component.

For that reason, this chapter treats partner integration controls as part of the security system, not as a commercial appendix.

In short: **Chapter 6 is the technical trust contract for Stablenotes v1**. If these requirements are met and evidenced, the product is deployable. If they are not, scale should stop.

Chapter purpose and scope

Purpose: define the security, reliability, and integration controls that make Stablenotes v1 deployable as an SDK-first recovery layer.

Scope includes:

- key generation quality (RNG)
- key non-retention / memory hygiene
- print survivability and legibility
- tamper evidence (v1 baseline)
- partner integration security boundary and liability model

Cross-reference note:

- Smart contract security details live in **Chapter 5 (Smart Contract Specification)**
- Regulatory positioning lives in **Chapter 11 (Regulatory Positioning)**
- Roadmap / v2 gates live in **Chapter 12 (Roadmap & Gates)**

6.1 Security model overview

6.1.1 Security objectives

- Unpredictable key generation
- No secret retention
- Durable recoverability
- Credible tamper evidence (v1 scope)
- Safe partner integrations at scale

6.1.2 Threat model summary

- Device/app compromise
- Accidental leakage (logs, telemetry, crash dumps)
- Physical degradation
- Tamper attempts
- Partner integration errors

6.1.3 v1 vs v2 assurance boundary

- **v1:** owner-controlled cold storage / recovery
- **v2:** bearer-transfer claims require additional adversarial testing (future-gated)

Cross-references:

- v2 adversarial requirements → **6.5.6** and **Chapter 12**
- Partner responsibilities → **6.7**

6.2 RNG and Key Generation Assurance

6.2.1 Purpose and scope

This section defines the minimum requirements for random number generation (RNG) used to create private keys in the Cryptocash wallet app, SDK and any companion

software. The objective is simple: **private keys must be unpredictable, unique, and non-recoverable by any party other than the user**.

A weak RNG invalidates the entire security model, regardless of device certification, printing quality, or tamper evidence.

6.2.2 Security objectives

The system must ensure that generated private keys are:

- **Cryptographically random** (high entropy, no predictable patterns)
- **Unbiased** (no systematic skew in output)
- **Non-repeating** (no duplicate keys across devices/sessions)
- **Not retained** (no logging, caching, or persistent storage outside intended user-controlled output)
- **Safely erased from memory** after use

6.2.3 Implementation requirements

6.2.3.1 Approved RNG sources (OS CSPRNG only). The app must use the platform cryptographic RNG provided by the operating system.

- **iOS:** use the system cryptographic RNG API (SecRandomCopyBytes)
- **Android:** use the platform cryptographic RNG (SecureRandom) through standard crypto APIs

6.2.3.2 No manual seeding. The app must not:

- seed RNGs manually using timestamps, device IDs, usernames, or any predictable values
- mix user-provided entropy in a way that replaces or degrades OS RNG behavior

6.2.3.3 Isolated keygen path. Private key generation must occur in a dedicated code path with no analytics calls, no debug logging, no crash-report payload inclusion, no clipboard use and no background task handoff.

6.2.3.4 Immediate post-generation handling. After generation, key material must:

- be used only for the intended workflow (e.g., note creation / recovery media encoding)
- be verified (QR + fallback text/checksum validation)
- be cleared from memory as soon as the workflow is complete or aborted

6.2.3.5 Secure erase behavior definition. The implementation must define and document:

- which buffers may contain key material

- how those buffers are zeroized
- when zeroization is triggered (success, failure, cancel, crash recovery path)

Note: “Secure erase” in application memory is best-effort and platform-constrained; this limitation must be disclosed.

6.2.4 Prohibited practices

- Custom RNG or manual RNG seeding
- Logging key material or derived secrets
- Persisting keys in app storage “temporarily” (Temporary persistence)
- Sending private keys to backend services
- Running key generation in instrumented/debug builds outside controlled test protocols
- Using clipboard for private key transport

6.2.5 Standards alignment

The RNG design and assurance process should be documented in terminology aligned with:

- **NIST SP 800-90A/B/C** (DRBG, entropy source, construction)
- **FIPS 140-3** concepts for cryptographic module assurance (where applicable)
- **ISO/IEC 18031** (random bit generation)
- **BSI AIS 20/31** (RNG evaluation guidance, especially relevant in Europe)

Full certification is not required at launch, but the design should be explainable against these frameworks.

6.2.6 Validation and testing

- Independent code review
- Runtime instrumentation
- Statistical sanity checks (supporting only)
- Duplicate-key monitoring

6.2.7 Acceptance criteria (ship gate)

- Explicit pass/fail requirements

6.2.8 Evidence artifacts

- RNG design, code review, non-retention tests, duplicate-key SOP

Cross-references:

- Secret handling lifecycle → **6.3**
- Partner integration controls for SDK use → **6.7.4**

6.3 Key Non-Retention and Memory Hygiene

This section defines the requirements for ensuring that private keys and other secret material are **not retained** by the Cryptocash wallet app, SDK, or companion device software beyond the minimum time necessary to complete the intended user workflow.

This is a core security control. Even with strong RNG, a retained key (in logs, storage, telemetry, crash dumps, or memory remnants) can result in total fund loss.

6.3.1 Security objectives

Private keys and secret material must be handled as **ephemeral data**. The system must ensure that secrets are created only when needed, exist in memory for the shortest possible time, are never written to persistent storage unless explicitly required by the user-controlled security model, are never included in logs, telemetry, analytics, crash reports, or debugging tools and are zeroized or invalidated immediately after use (subject to platform constraints).

6.3.2 Threats addressed

This control mitigates accidental leakage via app logs, crash-dump leakage, telemetry/analytics exfiltration, temporary-file leakage, clipboard leakage, background task or inter-process communication (IPC) leakage and memory scraping on compromised devices (partially mitigated; not eliminated).

6.3.3 Data classification model

- **Class A** (critical secrets such as private keys, mnemonics/seed phrases, raw entropy bytes, unencrypted key export buffers)
- **Class B** (sensitive non-secret such as public addresses, transaction metadata and note identifiers (if not directly linkable to private keys)
- **Class C** (operational such as: UI events, performance metrics, non-sensitive diagnostics)

6.3.4 Implementation requirements

No persistent storage. Class A secrets must not be stored in local files, shared preferences / UserDefaults, databases, caches, analytics payloads, nor in backups (cloud or device-level app backups). If a user workflow requires persistence (e.g., explicit wallet backup), it must be user-initiated, clearly disclosed and protected by separate controls (covered elsewhere).

No secret logging. The application and SDK must not log raw private keys, mnemonics, entropy, encoded key payloads, QR contents containing secrets. This

applies to production logs, debug logs, analytics events, error traces and third-party SDK telemetry.

Memory minimization. Secret material must be allocated in as few buffers as possible, copied as few times as possible, kept in scope for the shortest possible duration. Developers must avoid string conversions of secrets (immutable strings are hard to sanitize), unnecessary serialization/deserialization, passing secrets through generic logging or exception wrappers

Zeroization / invalidation. Where platform/language permits, buffers containing Class A secrets must be actively cleared after use. Documentation must specify which buffers are cleared, by what method and at what lifecycle event (success, cancel, error).

Where secure zeroization is not fully guaranteed (e.g., managed runtimes), the implementation must minimize copies, use mutable buffers, reduce object lifetime and explicitly disclose residual risk.

Clipboard prohibition. Class A secrets must never be copied to or through the system clipboard in production workflows.

Background/interrupt behavior. If the app is backgrounded or interrupted during a sensitive workflow, then active secret buffers must be invalidated or cleared, sensitive screens must be redacted from app switcher previews (where supported) and the workflow must require explicit user re-entry or restart.

Crash reporting controls. Crash reporting and analytics systems must be configured to redact memory dumps (where configurable), exclude sensitive context, disable custom breadcrumbs in sensitive workflows and maintain an allowlist of approved telemetry fields (Class C only).

IPC / bridge restrictions. Secrets must not be passed through deep links, URL schemes, push notifications, webviews, JavaScript bridges and untrusted IPC channels.

6.3.5 SDK-specific partner requirements

The SDK must provide a documented “secret-safe” API surface (no APIs that encourage persistence/logging), integration guidance for partner apps on telemetry redaction and crash configuration and a compliance checklist for partners to confirm they are not retaining secrets in their own layers.

This is essential for an SDK-first option, because partner integrations can reintroduce leakage even if Stablenotes code is clean.

6.3.6 Validation and testing

Static analysis and code review. Independent review must verify that there are no secret logging paths, no persistence paths for Class A data, no clipboard or unsafe IPC use and that controlled lifecycle of secret buffers is in place.

Runtime leakage tests. Controlled test builds need to be implemented to verify secrets do not appear in local storage, caches, crash reports, analytics payloads and debug consoles.

Crash-path testing. Crashes need to be simulated during key-generation/printing workflows to confirm that secrets are not exposed in captured reports and that resumed app state does not contain recoverable key material.

Background/interrupt testing. Simulation is needed for app backgrounding, incoming calls, OS task switching and low-memory events.

Third-party SDK audit. Inventory of all third-party SDKs needs to be made to verify none can capture Class A data via auto-screen capture, session replay, verbose error logging and analytics parameter scraping.

6.3.7 Acceptance criteria (ship gate)

Non-retention and memory hygiene controls are production ready only if independent review confirms no secret persistence or logging paths, runtime tests show no Class A leakage in storage, telemetry, or crash reports, interrupt/crash-path behavior is tested and documented, third-party SDK risk review is completed and any unsafe SDKs are removed or sandboxed and partner integration guidance (for SDK customers) is published

6.3.8 Evidence artifacts

- Classification policy, code review, leakage tests, SDK guide

Cross-references:

- RNG source and key creation → **6.2**
- Partner certification requirements → **6.7.4**
- Incident response classification → **6.7.5**

6.4 Print Survivability and Key Legibility

This section defines the requirements for ensuring that printed recovery media (QR code + human-readable backup text) remains **readable and recoverable** over the intended service life under normal handling and storage conditions.

This is a core security requirement. If a private key becomes unreadable, funds are effectively lost even if no attacker is involved. Therefore, is also extremely important that no other substrate is used except authorized notes shipped along with the device and offered as an after-market consumable.

6.4.1 Purpose and scope

Ensure printed recovery media remains recoverable over intended service life and ensure that only genuine notes are used for the imprints as this narrows down the critical combination ink + substrate (ink properties can change dramatically over different substrates).

6.4.2 Security and reliability objectives

Scannable, legible, stable, creation-time verifiable, batch-consistent

6.4.3 Threats and failure modes addressed

Smudging / abrasion from handling, water damage / humidity swelling, UV fading / thermal degradation, chemical staining (sweat, sanitizer, beverages, detergents), adhesive migration from tamper stickers, low-contrast or misaligned prints, QR corruption from creases/folds, “Looks fine but won’t scan” failures (poor scan behavior), printing in other media than the authorized.

6.4.4 Media design requirements

Dual-format recoverability (QR + fallback text). After imprint, each note must contain a **machine-readable QR code** carrying the required recovery payload, a **machine-readable matrix code** pre-printed at factory level and linked to the note’s unique serial number using a secret security layer and a **human-readable fallback** representation with checksum/error-detection.

The human-readable fallback must be sufficient to recover funds if the QR code is unreadable, but the text remains legible.

QR constraints (ECC, module size, quiet zone, contrast). The QR format must define and fix error correction level (ECC), minimum module size, quiet zone margin, contrast ratio target, and placement tolerances by means of using a guidance ruler.

The QR design must be validated against a supported device matrix (camera quality + OS versions).

Fallback text constraints (checksum, readability). Fallback text must include checksum or integrity marker, typography and spacing optimized for readability, contrast and font size minimums AND rules for avoiding ambiguous characters (e.g., O/0, I/l)

Redundancy requirements. The system should implement at least one redundancy mechanism, such as duplicate QR in separate area, segmented encoding, redundant text blocks, recovery code split with checksum. If no redundancy is used, the rationale and risk acceptance must be documented.

Tamper-sticker compatibility. Adhesive chemistry and sticker coating must be qualified for long-term compatibility with the ink and substrate. Adhesive migration must be characterized. The sticker must not reduce readability below acceptance thresholds over service life. Opening/removal behavior must be characterized (tearing, residue, visibility).

6.4.5 Printing process requirements

Controlled print pipeline. The printing process must define the printer model(s) and firmware versions (or approved classes), print resolution and calibration settings, media handling conditions (temperature/humidity range) and consumable specifications (inkjet cartridge lot controls, note batch controls).

Post-print verification gate. Before any irreversible step (e.g., key finalization/erase), the workflow must require QR scan verification on-device along with note scan verification, and human-readable fallback integrity check (checksum or format validation).

If verification fails, the note must be rejected and destroyed.

Batch QA and rejection logging. Production and pilot printing must include batch sampling, print contrast checks, alignment checks, scan-rate checks and rejection logging by defect type.

6.4.6 Durability test program

The durability qualification program must follow as a minimum the **DIN ISO 14145** Standard. It was originally developed for long-term archivability of documents (Part 2: documentary use DOC) containing text and signatures done with roller ball pens (and corresponding refills) but it can be considered highly relevant for the purpose of qualifying the note product and imprint process.

Samples must be tested in a standard atmosphere 23/50 in accordance with **DIN EN 20187**, i.e. at an ambient temperature of (23 +/-1) ºC and a relative humidity of (50 +/-2) %.

Mechanical wear. Includes erasure resistance, dry rub / abrasion, repeated handling, folding / crease cycles and edge wear

Moisture / water. Test samples must be exposed to splash and wiped after moisture exposure. Test must include humidity cycling and short-duration water exposure or water resistance.

Thermal / UV aging. Includes elevated temperature exposure, temperature cycling, UV exposure / light aging and lightfastness up to level 5.

Chemical exposure. At minimum test is needed against common-contact substances: skin oils / sweat simulation, hand sanitizer, detergent/soap residue and common beverages (water, coffee, soda). Standarized **DIN ISO 14145** test rounds should include ethanol resistance, hydrochloric acid resistance, ammonium hydroxide resistance and bleaching resistance.

Sticker/adhesive aging. adhesive migration, edge lift, residue formation and interaction with ink and substrate under heat/humidity aging

6.4.7 Acceptance criteria (ship gate)

The print survivability system is production ready only if all following conditions are met:

Creation-time verification pass. 100% of issued notes pass QR + fallback verification at creation time and failed prints are rejected and logged.

Post-stress QR readability threshold. After durability tests, QR scan success must meet a defined threshold across a supported device matrix (e.g., threshold set by product requirements and test protocol).

Post-stress fallback legibility threshold. After durability tests, fallback text must remain legible and checksum-valid at the defined threshold.

Batch consistency. QA sampling demonstrates print quality remains within tolerance across lots and devices.

Sticker compatibility threshold. No adhesive or tamper component may reduce readability below threshold during the claimed service life.

6.4.8 Supported device matrix

Because smartphone camera performance varies, the company must maintain a supported scanning matrix covering:

- representative Android devices (low/mid/high tier)
- representative iPhone models
- supported OS versions
- typical lighting conditions

The matrix must be reviewed periodically and updated as devices/OS versions change.

6.4.9 User handling guidance and replacement policy

The product includes plain-language guidance covering:

- why to avoid printing in substrates other than the supplied notes
- how to store the note
- what to avoid (moisture, direct heat, chemicals, abrasion)
- what to do immediately if damage occurs
- how to verify readability periodically (optional recommended check)
- when to replace/re-issue the recovery media

On top of the handling guidance, a replacement policy must be clearly defined and documented:

- replacement procedure for damaged but still-recoverable notes
- support process for partial legibility failures
- escalation path for “unreadable note” incidents
- whether any warranty applies (and its limits)

This policy is part of product safety, not just customer support.

6.4.10 Validation and testing evidence

Independent lab report. An external lab must test the defined matrix and report conditions, sample sizes, failure rates, scan/legibility outcomes, pass/fail against acceptance criteria and requirements met or not met.

Internal QA validation. Internal QA must demonstrate repeatable print quality across batches, defect classification plus rejection workflow and traceability of materials and consumables

Field validation cohort (recommended before scale-up). A controlled field cohort must be executed to capture real handling behavior, scan failures by device model, environmental failure patterns and replacement rates.

6.4.11 Evidence artifacts

- Media spec, durability protocol, lab report, QA report, user guide

Cross-references:

- Tamper-layer compatibility → **6.5.4**
- Partner support responsibilities → **6.7.3** and **6.7.5**

6.5 Tamper Evidence and Adversarial Access

6.5.1 Purpose and scope

This section defines the requirements for **tamper evidence** and resistance to **adversarial access attempts** against printed recovery media and any tamper-evident resource (e.g., security sticker).

The goal is to clearly separate:

- **v1 (Cold Storage):** minimum credible tamper-evidence claims for owner-controlled recovery media
- **v2 (Bearer/Transfer use):** much stronger adversarial-transfer claims required before any transactional/bearer product launch

This prevents over-claiming in v1 while preserving a rigorous path to v2.

The use of a tamper-evident security sticker already during v1 (Cold Storage) might be considered overkill since the recovery media is fully owner-controlled and the risk of fraud under such conditions is practically non-existent provided the notes are stored safely. However, the introduction of a security sticker already during v1 fulfils two main objectives:

a) It provides visible evidence that the value on the note has been redeemed, recovered or converted. Thereby the note should be considered as empty and segregated from other notes where the security sticker is still closed.
b) It enables the deployment and testing of tamper-proof characteristics of the security sticker under real life conditions and on a bigger scale before moving forward to a transactional product (v2).

The scope applies to printed recovery media carrying secret material (QR and/or text), tamper-evident overlays (security sticker that includes VOID features and obfuscation layers), user verification cues (visual indicators, border text, tear evidence), packaging and handling steps that affect tamper evidence, adversarial testing protocols and acceptance thresholds

The scope covers unauthorized viewing attempts, partial opening / resealing, label replacement / swap, stamp off secret material to other surface (boiled egg attack),

optical extraction attempts, residue and edge-lift attacks and counterfeit visual mimicry (v2 scope primarily).

6.5.2 Security objectives

The tamper system must provide **defined, testable tamper evidence**, not vague "anti-counterfeit" claims. At a minimum (v1), it must aim to ensure that secret material is not plainly readable before intended opening, that opening causes visible, irreversible evidence under normal inspection, that tamper evidence remains functional over the expected service life and that users can follow a simple verification checklist before recovery.

For v2 (future), the bar is higher:

- Practical read-without-open and reseal attacks are not feasible at meaningful rates
- Field users can detect tampered notes reliably using simple procedures

6.5.3 v1 vs v2 claim boundary (critical)

v1 is owner-controlled recovery media, not a bearer instrument. Tamper evidence in v1 supports **owner confidence**, not anonymous circulation.

- **v1 allowed claims:** tamper-evident, obscures secret until opened, visible opening evidence under defined conditions.
- **v1 prohibited claims:** counterfeit-proof, unforgeable, safe for bearer transfer, prevent all copying.

v2 — Bearer / Transfer (future-gated). v2 requires materially stronger claims and independent adversarial testing against hostile transfer scenarios. These claims should not appear in v1 marketing or deployment materials until validated.

- **v2 claims:** only after adversarial testing and field validation

6.5.4 Threat model (tamper-specific)

The system design and tests must consider the following attacker classes:

A) Casual attacker
Attempts basic peeling, edge lifting, steaming, or light-based viewing using consumer tools.

B) Skilled attacker
Uses magnification, controlled heating, solvents, optical enhancement, micro-lifting, and resealing attempts.

C) Organized attacker (v2-critical)
Performs repeated tests with specialized tools, custom labels, and counterfeit reproduction workflows.

6.5.5 Material and design requirements (v1)

Obfuscation effectiveness (pre-open). When the sticker is used, the secret payload must be visually obscured or obfuscated prior to opening under normal viewing conditions. The design must define viewing conditions (lighting, angle), acceptable visibility threshold and prohibited leak-through (e.g., characters visible through obfuscation).

Irreversible opening evidence. Opening the tamper layer must create visible, irreversible evidence such as VOID pattern, edge text disclosure, delamination pattern and substrate damage / residue pattern. The evidence must be observable without specialized tools.

User verification cues. The product must include a simple “before opening” and “after opening” visual checklist including intact border alignment, intact tamper marks, no edge lift / bubbles / residue anomalies and expected opened-state pattern visible after opening.

Shelf-life compatibility. Tamper-evident behavior must remain functional over the claimed storage life so that the sticker must not self-lift, the obfuscation must not degrade and reveal text and so that the adhesive must not destroy print readability during normal aging.

Opening workflow guidance. The recovery workflow must assume the note becomes higher-risk after opening: user should recover funds promptly, opened notes should be treated as compromised for future long-term storage, guidance must instruct users not to re-store opened notes as “secure” and segregate them from closed notes.

6.5.6 v1 test program (mandatory pre-launch)

The v1 program must test edge lifting attempts (using manual tools), peel attempts under normal conditions, oblique-light viewing attempts, basic heat/humidity exposure impact on tamper behavior, adhesive aging and residue patterns, user inspection clarity (can normal users tell opened vs intact?).
The output should establish a **baseline tamper-evidence claim** only.

6.5.7 v2 adversarial test program (future-gated)

The v2 program must include independent red-team rounds covering controlled optical extraction attempts, chemical/thermal opening and reseal attempts, counterfeit sticker reproduction, note/sticker swap scenarios and field deception tests against actual user/agent SOP.

Before any bearer/transfer product launch, the system must additionally demonstrate:

Read-without-open resistance. Attackers cannot practically recover the secret without triggering visible tamper evidence.

Reseal resistance. Attackers cannot open, copy, and reseal in a way that passes field inspection at meaningful rates.

Swap/counterfeit resistance. Attackers cannot substitute a counterfeit or altered note/sticker that passes the lay verification SOP at meaningful rates.

Field deception tests against lay SOP. Non-expert users (or agents) can detect tamper/counterfeit indicators reliably using a short checklist.

6.5.8 Acceptance criteria

- **v1 ship gate:** baseline tamper evidence consistency + usability
- **v2 gate:** independent adversarial success rates below threshold

6.5.9 User guidance requirements

The product must include tamper guidance:

- How to inspect the note before opening
- What "opened/tampered" looks like
- What to do if tamper is suspected
- Why opened notes should not be stored long-term
- How to safely proceed with recovery after opening

This reduces support burden and prevents misuse.

6.5.10 Validation and evidence artifacts

Material compatibility report. Confirms sticker/adhesive/ink/substrate compatibility over aging and environmental conditions.

Tamper baseline test report (mandatory for v1). Documents attack methods attempted, sample sizes, visible outcomes, consistency of opening evidence and observed failure modes.

User inspection usability test report (mandatory for v1). Confirms users can identify intact vs opened samples using the provided checklist.

Independent adversarial red-team reports (mandatory for v2 only). Required before any bearer/transactional claims or pilots.

Cross-references:

- Print/adhesive compatibility → **6.4.4** and **6.4.6**
- v2 roadmap gates → **Chapter 12**
- Partner support scripts and claims control → **6.7.5** and **6.7.7**

6.6 Serialized Note/Device Supply Chain and Distribution Assurance

It should include:

- serialization requirements
- manifest and chain-of-custody controls

- warehouse and retail controls
- exception management
- audit evidence

6.6.1 Purpose

Define a traceable, tamper-evident, low-complexity supply chain for Stablenotes note media (as an aftermarket consumable) and Cryptocash ATM device from factory → warehouse → distribution points → retail issuance.

The goal is not to copy a central bank. The goal is to borrow the useful parts such as serialization, chain-of-custody, tamper-evident packaging, reconciliation and exception handling.

That materially improves security without creating an unscalable cost structure.

6.6.2 Security objectives

This supply chain should support five concrete objectives:

1. **Authenticity**
   Only genuine Stablenotes media and devices enters partner channels.
2. **Traceability**
   Every note/device and every batch can be traced by serial number and batch ID.
3. **Tamper detection**
   Any carton/pallet opening, substitution, or missing inventory is detectable quickly.
4. **Containment**
   If a batch is compromised, stolen, or defective, it can be quarantined by serial range.
5. **Operational accountability**
   Every custody handoff is attributable to a person/entity and timestamped.

6.6.3 Asset classes

It is important for risk and cost to differentiate notes and devices.

Class N — Serialized blank recovery media (default v1)

- Pre-printed note media with unique serials
- No stored value at shipment
- Primary risk: counterfeit stock, quality drift, unauthorized substitution
- First to scale

Class V — Serialized vouchers / pre-funded samples / Cryptocash ATM devices (higher risk)

- Notes carrying promotional value (e.g., small USDT voucher)
- Cryptocash ATM devices with starter-packs of blank notes inside
- Primary risk: theft, fraud, abuse, consumer disputes
- Requires tighter controls than Class N
- Tightly controlled marketing pilot lane only

6.6.4 Simplified “banknote-like” supply chain model

6.6.4.1 Factories (Approved printing house and device manufacturing facility)

Required controls for notes

- Each note gets a **unique note serial**, a **batch serial** and an optional lot code for security sticker.
- Printing house generates a **digital batch manifest** containing a batch ID, note serial range(s), production timestamp, material lots used and operator line / machine ID.
- Manifest is **signed** (digitally) and transmitted to Stablenotes.
- Defective/rejected notes are logged and destroyed with count reconciliation.

Packaging rules for notes

- Notes packed into **serialized inner packs** (e.g., 25/50/100)
- Inner packs packed into **serialized cartons**
- Cartons sealed with **tamper-evident tape/label**
- Pallets (if used) get pallet ID and pallet manifest

Required controls for devices

- Each device comes with a **read-only 96-bit unique device serial** (MCU ID) that cannot be altered by the user. During QA process, the MCU ID is printed in packaging, in certification sticker on device and software-readable via telemetry.
- Device manufacturing facility generates an entry into a **device database** containing MCU ID, production timestamp and operator ID.
- Stablenotes has authorized access to device database and can use it for device traceability throughout the supply chain

Packaging rules for devices

- A **serialized note starter-pack** (25 pcs) is packed into a **serialized device pack**
- Individual device packs packed into **serialized cartons** (8 pcs each)
- Cartons and device packs are sealed with **tamper-evident tape/label**

- Pallets get pallet ID and pallet manifest

Why this helps security
If counterfeit or defect issues emerge, isolation is possible by MCU ID, note serial range, batch, material lot and machine line.

6.6.4.2 Factory → central warehouse transport

Required controls

- Shipment created against a digital manifest (carton IDs + batch IDs)
- Handoff scan at origin (carton/pallet serials)
- Sealed transport (tamper-evident seal ID recorded)
- Handoff scan at destination
- Seal integrity check at receipt
- Exception workflow for seal mismatch, missing or damaged carton and manifest mismatch

"Simplified banknote-like" rule
Armored cars for Class N are not necessary. Instead, serialized scanning, seal verification and exception logging should be implemented.
For Class V vouchers and devices, tighter transport controls (insured courier, higher verification, smaller consignments) should be implemented.

6.6.4.3 Central warehouse (vault-lite, not central bank)

This is the core control point for the entire Stablenotes supply chain.

Required controls

- Segregated storage zones for Class N stock, Class V stock, quarantined stock and returned stock
- Access control: limited authorized staff, role-based access and entry logs
- Inventory system tracks: note serial range by location, carton status (received / picked / shipped / quarantined), batch and lot metadata
- Cycle counts: daily (high-value / Class V), weekly (Class N active SKUs), monthly full reconciliation
- CCTV coverage for receiving, picking, packing, quarantine

Dual control (should be kept simple)
Dual sign-off will be enforced only for: opening factory cartons, repacking stock, handling Class V voucher stock/devices and quarantine releases.
No need to dual-control every pick for blank media.

6.6.4.4 Warehouse → distribution points (partner DCs / field distributors)

Distribution points can be partner-operated warehouses**,** regional distributor hubs**,** authorized retail resupply points or remittance partner back offices

Required controls

- Ship by serialized carton/pack only (avoid loose notes)

- Scan-out at warehouse, scan-in at destination
- Destination acknowledges might include MCU IDs, carton IDs, tamper status and quantity
- Stock state transitions recorded: In Warehouse → In Transit → At Distribution Point

Exception handling
Any missing MCU IDs, serials or opened cartons would trigger immediate quarantine of affected batch, investigation ticket and hold on further shipment of same batch until cleared (if needed)

6.6.4.5 Distribution point → retail point / end channel

This is where most sloppy supply chain systems fail.

Required controls

- Retail receives **sealed packs**, not loose notes or devices
- Retail scans pack serial(s) and device MCU IDs into inventory
- Packs are opened only at issuance point (or controlled prep station)
- Unsold stock remains in sealed packs where possible
- End-of-day reconciliation must include MCU IDs, issued serials, remaining serials and damaged/voided serials

Critical rule
No "informal" stock transfers between retail points. All transfers must be scanned and recorded.

6.6.5 Retail distribution strategy

Following an SDK-strategy first, retail should be **partner-led and controlled**, not mass shelf or open-shelf retail at launch.

6.6.5.1 Phase 1 retail strategy

Channel type

Use **authorized counter distribution** through partners:

- exchange storefronts / partner service points
- remittance agent locations
- electronics/mobile shops with trained staff
- partner branches (where applicable)

Why counter distribution first

- better staff control and stock reconciliation
- fewer theft/substitution incidents
- easier user education (handling + tamper checks)

6.6.5.2 What is sold at retail

For v1 (Class N)
Devices and sealed serialized refill note packs (larger quantities for repeat users):
Optionally co-branded partner packaging (if partner-integrated)

For Class V vouchers (if used)
They should be treated as a separate SKU with tighter controls and kept as a marketing experiment, not core distribution.

- very small denominations
- short expiry dates
- one-time redemption rules
- stricter issuance logging

6.6.5.3 Point-of-sale process (simple but controlled)

At retail issuance:

1. Staff scans pack serial / note serial
2. Inventory system marks note as Issued
3. User receives handling/tamper guidance
4. (Optional) App/SDK can validate note serial authenticity on first use

This gives end-to-end traceability:
Factory batch → warehouse → retail point → issued note serial

6.6.5.4 Returns and damaged stock

A formal loop is needed, or fraud will fill the gap.

Return categories

- **Unopened sealed pack** → return to stock after inspection
- **Opened but unused note** → quarantine or controlled resale (policy-defined)
- **Damaged note media** → mark serial Void, quarantine, destroy with record
- **Suspected counterfeit** → quarantine + incident escalation

Every destroyed note serial should be logged as Destroyed with reason code.

6.6.6 How distribution assurance fortifies Stablenotes security objectives

The serialized note/device supply chain and distribution assurance is not just an operations process. It directly improves security:

A) Anti-counterfeit and anti-substitution
Serial and batch traceability makes it harder to inject fake notes or fake devices unnoticed.

B) Faster incident response

If one batch of notes has print defects, adhesive drift, or tamper-layer issues, exact serial ranges can be quarantined. The same thing is valid for batch of devices.

C) Better partner trust
Partners will integrate faster if the following can be proved:

- where stock came from
- who handled it
- and what happens when something goes wrong

D) Better forensic capability
If a user reports a problem, several things can be traced including batch, retail point, shipment and material lot. That's how Stablenotes' supply chain can move from anecdotes to root-cause analysis.

E) Lower warranty and support cost
Most "mystery failures" become inventory/handling exceptions that can actually be diagnosed.

6.7 Partner Integration Security Boundary and Liability Model

This section defines the **security boundary**, **operational responsibilities**, and **liability allocation model** for SDK-first deployments of Stablenotes technology.

This is essential for an SDK-first model, the core technical risk is not only Stablenotes code — it is also **how partners integrate it**. A clean boundary prevents:

- security regressions caused by partner implementation choices,
- compliance ambiguity,
- and endless blame-shifting after incidents.

Core principle
Stablenotes is provided as a **security and recovery infrastructure layer**.
In an SDK-first deployment:

- **Stablenotes owns** the security and reliability of its SDK/device/media components.
- **Partner owns** the regulated financial service relationship (where applicable), app UX decisions outside the SDK boundary, and customer operations.
- **User owns** self-custody decisions and handling of recovery media, subject to disclosed limitations.

This must be reflected consistently in contracts, product docs, and support scripts.

6.7.1 Purpose and scope

- Define responsibilities, shared-risk interfaces, and liability allocation for SDK deployments

6.7.2 Security boundary definition

- **Stablenotes-controlled components.** Stablenotes is responsible for the security of:
    - SDK code and documented APIs
    - Device firmware and secure update chain
    - Print workflow inside the Stablenotes-controlled path
    - Media specifications (QR format, fallback encoding, survivability requirements)
    - Tamper-evidence design (v1 claim scope)
    - Reference implementation used for integration validation
    - Partner integration guidance and security requirements
- **Stablenotes-controlled components.** Stablenotes is responsible for the security of:
    - SDK code and documented APIs
    - Device firmware and secure update chain
    - Print workflow inside the Stablenotes-controlled path
    - Media specifications (QR format, fallback encoding, survivability requirements)
    - Tamper-evidence design (v1 claim scope)
    - Reference implementation used for integration validation
    - Partner integration guidance and security requirements

- **Stablenotes-controlled components.** Stablenotes is responsible for the security of:
    - SDK code and documented APIs
    - Device firmware and secure update chain
    - Print workflow inside the Stablenotes-controlled path
    - Media specifications (QR format, fallback encoding, survivability requirements)
    - Tamper-evidence design (v1 claim scope)
    - Reference implementation used for integration validation
    - Partner integration guidance and security requirements

These areas require integration certification before production deployment.

6.7.3 Responsibility model (RACI)

- Stablenotes obligations (SDK security, docs, advisories, evidence)
- Partner obligations (integration correctness, telemetry controls, support training, patching)

6.7.4 Integration security requirements (production gate)

- Golden path requirement
- Telemetry/logging controls
- Version and patching SLAs
- Integration certification (functional + security + ops)

6.7.5 Incident response model

- Incident categories (Stablenotes / Partner / Shared)
- Notification and triage SLAs
- Kill-switch / containment procedures
- Customer communications ownership

6.7.6 Liability allocation framework

- Liability follows control
- Warranty scope and exclusions
- Liability caps and carve-outs
- Consumer-loss policy framework (no blanket guarantee)

6.7.7 Evidence sharing and audit rights

- Evidence package obligations
- Partner attestations
- Audit/review rights

6.7.8 Branding and claims control

- Approved v1 claims
- Prohibited claims (cash equivalent, unhackable, guaranteed recovery)

6.7.9 Production readiness checklist (partner launch gate)

- Certification complete
- Logging controls verified
- Runbooks approved
- Contracts executed
- SDK version current
- KPI rollback thresholds agreed

6.7.10 Evidence artifacts

- Boundary spec, RACI, certification checklist, incident runbook, liability framework

Cross-references:

- Secret handling requirements for partner apps → **6.3.6**
- Print/tamper support guidance → **6.4.9** and **6.5.8**
- Regulatory boundary → **Chapter 11**

6.8 Security and Assurance Ship Gates Summary

6.8.1 v1 production ship gates (consolidated)

A single table summarizing required pass/fail criteria from:

- **6.2 RNG**
- **6.3 non-retention**
- **6.4 survivability**
- **6.5 v1 tamper evidence**
- **6.6 partner integration certification**

6.8.2 v2 future gates (separate, non-blocking for v1)

- Adversarial tamper testing
- Bearer-transfer field SOP validation
- Additional liability/reserve and compliance gating

Cross-references:

- Detailed acceptance criteria in each subsection

- Roadmap sequencing in **Chapter 12**

6.9 Claims-to-Evidence Mapping and Document Control

6.9.1 Claims-to-evidence matrix reference

- Link to the master matrix (spreadsheet/data room)
- State versioning and update cadence

6.9.2 Evidence artifact index

- Pointer list to all reports/documents referenced in 6.2–6.6

6.9.3 Change control

- Any change affecting security claims, print specs, or SDK integration requirements requires:
    - version bump
    - regression review
    - partner notification (if applicable)

7 FIRMWARE, APP, AND UPDATE INTEGRITY (STABLENOTES V1)

This chapter defines the minimum security architecture and operational controls for the **printer unit firmware**, its **update mechanism**, and the **Cryptocash wallet companion app** within Stablenotes v1. The intent is to meet "high assurance" expectations commonly applied to security-critical devices and software supply chains, including platform firmware resiliency guidance (NIST SP 800-193), secure software development practices (NIST SSDF), and supply chain risk management principles (NIST SP 800-161). (NIST Computer Security Resource Center)

Scope note: This chapter applies to the Stablenotes v1 printing device and v1 companion app role. Any consumer "banking-like" features (buy/sell/swap/card) are showcased in the v1 companion app just as pure reference for v1 partner deployments, particularly neobank partners which might want to see a fully integrated reference deployment.

7.1 Firmware architecture

7.1.1 Hardware components (reference architecture)

The printer unit SHOULD be treated as a security appliance. A typical secure architecture includes:

- **Hardware root of trust:** immutable boot ROM / secure bootloader
- **Secure element (or equivalent):** secure key storage and cryptographic operations *(placeholder: model/type)*
- **Main MCU/FPGA:** runs device firmware and print orchestration
- **Non-volatile storage:** firmware slots + configuration (encrypted/verified)
- **Print engine:** printer controller, sensors, and calibration
- **Connectivity module(s):** Wi-Fi only in order to minimize surface area. USB C interface is used only for charging.
- **Physical debug interfaces:** JTAG/SWD/UART *(must be locked/disabled in production)*
- **Device identity:** unique device certificate/keypair provisioned at manufacture *(placeholder)*

7.1.2 Secure boot and chain of trust (MUST)

The device MUST implement a **full chain of trust** from immutable hardware to the running firmware image:

1. **Boot ROM** verifies the first-stage bootloader using a vendor root key.
2. Bootloader verifies the **firmware image** signature (and optionally a measured hash).
3. Firmware verifies critical configuration and any loaded modules before use.

This “verify-next-stage-before-executing” model aligns with platform secure boot chains used in modern ecosystems (e.g., Apple secure boot chain of trust; Android Verified Boot). (Apple Support)

7.1.3 Firmware resiliency (MUST)

Firmware MUST be designed for **protect / detect / recover** behavior consistent with platform firmware resiliency guidelines: (NIST Computer Security Resource Center)

- **Protect:** prevent unauthorized modification (signed firmware, locked storage, debug disablement).
- **Detect:** detect corruption/unauthorized changes (signature validation; measured boot optional).
- **Recover:** support recovery to a known-good state (A/B slots, recovery partition, or factory-safe mode).

7.1.4 Secrets handling in firmware (MUST)

- Private key material and sensitive payloads MUST be treated as **ephemeral** unless explicitly required (e.g., device identity keys in secure element).
- Firmware MUST NOT log or persist private keys or printed recovery payloads.
- Sensitive buffers MUST be zeroized after use where feasible.
- Debug logs MUST be compiled out or restricted to non-sensitive operational codes in production builds.

7.1.5 Production hardening (MUST)

- Debug interfaces MUST be disabled or locked (fuses/secure element policy) for production units.
- Factory test modes MUST require authenticated entry and be disabled post-provisioning.
- Device must implement rate limits / lockouts for any privileged maintenance commands.

7.2 Updates

7.2.1 Signed updates (MUST)

All firmware updates MUST be **cryptographically signed** and verified on-device before installation.

Minimum requirements:

- Signature verification occurs **before** writing to the active boot slot.
- Update packages are authenticated and integrity-checked end-to-end.

- Update signing keys are protected by an HSM-backed process and role separation.

7.2.2 Key management and signing hierarchy (MUST)

Stablenotes SHOULD adopt a structured update signing model similar in spirit to modern update security frameworks (e.g., TUF concepts: separation of roles, survivability under partial key compromise). (TUF)

Minimum key management requirements:

- **Root key(s):** offline storage (HSM or air-gapped), used only for key rotations and critical policy updates.
- **Release signing key(s):** used for routine firmware releases; protected by HSM; rotated on schedule.
- **Threshold policy:** for high-risk releases, require M-of-N approvals (two-person integrity).
- **Key ceremony:** documented creation, rotation, revocation, and incident response procedures.

7.2.3 Rollback protection (MUST)

The device MUST protect against downgrade attacks.

- Maintain a **minimum allowed firmware version** (monotonic counter or secure element-backed anti-rollback).
- Block installation of firmware below the minimum version.
- Permit rollback only to a **known-good** version within an approved window (if required for safety), never to a vulnerable baseline.

Android Verified Boot explicitly supports rollback protection concepts; the device should meet an equivalent standard appropriate to its platform. (Android Open Source Project)

7.2.4 Reproducible builds and provenance (SHOULD; target high assurance)

For partner and auditor trust, firmware and companion software SHOULD be built with strong supply chain provenance:

- **Reproducible or verifiable builds** (documented build pipeline, deterministic artifacts where feasible).
- **Build provenance attestation** targeting SLSA-style controls (e.g., tamper-resistant build service, signed provenance). (SLSA)
- **SBOM** generated for each release and retained (with dependency risk tracking).
- Secure SDLC aligned to NIST SSDF practices. (NIST Computer Security Resource Center)

7.2.5 Update rollout controls (MUST)

- Staged rollout (canary → limited → full) with measurable health signals.

- Emergency pause mechanism for failed releases.
- Mandatory update path for critical security fixes within defined SLA.

7.2.6 Supply chain integrity and manufacturing controls (high level; MUST)

Manufacturing and provisioning MUST implement baseline cybersecurity supply chain risk controls consistent with established C-SCRM guidance. (NIST Computer Security Resource Center)

At a minimum:

- **Secure provisioning stations** (access controlled, audited -relevant for v2 only).
- **Unique device identity** provisioned per unit; keys injected only under controlled procedures.
- **Firmware loaded only via authenticated process**; prohibit "side-loading" in factory.
- **Chain-of-custody** for devices and critical components (secure element lots, MCU lots).
- **Factory-to-warehouse tamper evidence** and reconciliation (see Chapter 6 supply chain assurance).

Evidence artifacts (examples):

- Firmware_Architecture_Spec_v1.pdf
- SecureBoot_and_AntiRollback_Design.pdf
- Update_Signing_Key_Management_Policy.pdf
- SLSA_Provenance_Attestation_v1.json *(or equivalent)*
- Firmware_SBOM_<version>.spdx.json
- Manufacturing_Security_Provisioning_SOP.pdf

7.3 Companion app (Cryptocash Wallet app) — v1 role and boundaries

7.3.1 Minimal v1 role (recommended for partner deployments)

For Stablenotes v1, the companion app SHOULD be scoped to a **recovery and verification interface**, but demonstrating integration capabilities towards common FinTech/DeFi functionalities.

Recommended minimal responsibilities:

- Initiate the "Create Recovery Note" flow via SDK
- Perform **post-print verification** (scan QR, validate checksum/fallback integrity)
- Display **handling and storage guidance**
- Provide **read-only monitoring** of public address state (optional)
- Provide **recovery/import** workflow when needed

7.3.2 What v1 explicitly does NOT do (non-negotiable for v1 scope)

In v1 deployments, the companion app and the Stablenotes SDK integration MUST NOT:

- Provide custody, hosted accounts, or key escrow
- Provide exchange execution, brokerage, or market-making
- Provide buy/sell/swap functionality as a PrintDreams-provided service
- Present itself as a bank, neobank, payment institution, or e-money provider
- Operate a cash-out network, operator program, or any “bearer note” transfer system

If buy/sell/swap/card functionality exists in a reference wallet for demonstration purposes, it must be clearly segmented as a **third-party regulated service** and treated as outside the v1 assurance scope. Partner deployments should default to disabling such features unless explicitly approved and contractually structured.

7.3.3 App integrity controls (MUST)

To support the security model in Chapter 6:

- App must enforce **non-retention** of secrets (no logging, no persistence, no clipboard use).
- Disable session replay / screen recording in sensitive flows.
- Restrict third-party analytics SDKs on key-handling screens.
- Implement hardened crash reporting (no memory dumps, redaction on sensitive flows).

7.3.4 OS trust anchoring (SHOULD)

When feasible, partner deployments SHOULD leverage platform security primitives:

- iOS secure boot chain of trust (baseline device integrity) (Apple Support)
- Android Verified Boot / integrity checks for supported devices (Android Open Source Project)

7.3.5 Implementation placeholders to complete before production

- Printer unit hardware bill of materials summary: **[TBD]**
- Secure element model and policy (for v2): **[TBD]**
- Secure boot key hierarchy + custody model: **[TBD]**
- Anti-rollback mechanism: **[TBD]**
- Update distribution model (direct OTA / partner-managed): **[TBD]**
- Companion app telemetry stack and allowed fields: **[TBD]**
- Partner integration certification checklist for app hardening: **[TBD]**

Evidence artifacts (examples):

- App_DataFlow_and_NonRetention_Test_Report.pdf
- ThirdPartySDK_Inventory_and_RiskReview.pdf
- SensitiveScreen_Redaction_and_NoReplay_Config.pdf
- Release_Signing_and_Distribution_SOP.pdf

8 PHYSICAL RECOVERY MEDIA DESIGN

This chapter specifies the physical recovery medium ("note") design for Stablenotes v1. The v1 objective is **recoverability and durability**, not bearer transfer. The design therefore prioritizes: (i) reliable decoding, (ii) redundancy against partial damage, (iii) clear user verification at creation time, and (iv) baseline tamper evidence without over-claiming.

Terminology: In v1, the "note" is a **recovery instrument**. It may contain highly sensitive recovery material. Loss or exposure of the secret may result in loss of funds.

8.1 Encoding format

8.1.1 Recovery payload definition (placeholder)

The recovery payload encodes the minimum data required to restore control of the Recovery/Note Wallet according to the v1 asset model.

Payload contents (TBD per chain and wallet model):

- chain_id or chain identifier (e.g., TON)
- wallet_type (single-key / contract wallet / other)
- secret_material:
  - **Option A (direct key):** private key (highest risk; simplest)
  - **Option B (seed):** seed phrase / entropy bytes (risk varies)
  - **Option C (encrypted secret):** secret encrypted under user-chosen passphrase (adds UX risk)
- checksum / integrity markers
- format_version
- Optional: note_id (non-sensitive), created_at timestamp, asset_label *(must not leak secrets)*

v1 recommendation: Use a versioned, canonical binary payload format (CBOR or protobuf-like structure) before QR encoding to ensure deterministic parsing and future extensibility. *(Placeholder: final format choice.)*

8.1.2 QR specification (recommended baseline)

QR codes must be specified as production requirements, not "best effort."
The relevant International Standards ISO/IEC 15415 and ISO/IEC 29158 must be followed strictly wherever possible.

QR type: Standard QR Code Model 2.

Error correction (ECC):

- **Recommended: ECC Level H (≈30% recovery)** for the secret QR containing the private key, because physical wear is expected.

- If payload size forces lower ECC, the trade-off needs to be documented explicitly.

Minimum module size (print constraint):

- **Recommended minimum: 0.5 mm per module** (for consumer smartphone cameras in realistic lighting).
- **Hard floor: 0.4 mm** only if verified by device-matrix testing.

Quiet zone:

- **Minimum:** 4 modules on all sides (QR standard baseline).
- **Recommended:** 6–8 modules where space allows to improve scan reliability under skew and low contrast.

Contrast requirements:

- **Target reflectance contrast: ≥ 40%** (practical scanability threshold), with a preferred higher margin depending on substrate.
- Glossy coatings over the QR area should be avoided unless tested.

Printing Placement rules:

- Clear indication of mobile ATM device outline on recovery notes for perfect positioning
- QR must not cross fold lines, perforations, or sticker edges.
- QR must be placed on a flat region with a defined “no-touch” handling zone where feasible.

Printing Placement Aid Accessory:

- User should be advised to purchase and use a placement aid accessory to perfectly align mobile ATM device with recovery note for best and consistent results.

Creation-time verification gate (mandatory):

- Every issued note must pass a post-print scan test (QR decode success + checksum validation) before finalization.

8.1.3 Human-readable fallback (format + checksum)

A fallback format must allow recovery even if the QR fails.

Fallback format (recommended):

- Base32 or Base58 encoding (ambiguous characters should be avoided).
- Grouped into fixed blocks for transcription accuracy, e.g.: XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX-XXXX
- Printed with high-contrast font, minimum size defined by readability tests.

Checksum / integrity markers (mandatory):

- At least one strong checksum that catches:
    - single-character errors,

  - transpositions,
  - partial block loss.

Recommended approach:

- Per-block checksum (small) + full-string checksum (strong), to support partial verification.

Verification:

- The app must validate fallback format integrity during post-print verification:
  - full checksum passes, and
  - block structure matches version.

8.2 Redundancy

8.2.1 Why redundancy is required

Without redundancy, a single failure mode can cause irrecoverable loss:

- a smudge removes a QR module region,
- a fold line destroys a few characters,
- water damage blurs a small area,
- a single character is lost in fallback text.

Principle: Redundancy converts “one defect = total loss” into “defect tolerated within defined limits.”

8.2.2 Redundancy options (choose one; document explicitly)

Option 1 — Primary QR + backup QR (recommended if space allows)

- Two identical secret QRs in separate physical locations on the same note: One in the front and one in the back.
- Each QR has independent survivability.

Pros: simplest; robust.
Cons: increases attack surface if tamper-evidence is weak; more space.

Option 2 — Segmented encoding (recommended for higher durability)

- Split payload into N segments (e.g., 3 of 5) using an erasure code (Reed–Solomon style).
- Each segment printed as its own QR (or mixed QR+text).

Pros: tolerates partial loss; best durability economics.
Cons: more complex decoding; requires app support.

Option 3 — QR + full fallback + partial fallback duplication (balanced)

- One QR (ECC H) + full fallback text

- Duplicate only the checksum and critical header/version fields in a second location.

Pros: reasonable compromise; simpler UX.
Cons: lower redundancy than full duplication.

Option 4 — Hidden redundancy in QR + large ECC + placement strategy (least preferred)

- Rely on ECC + careful layout only.
- Accepts higher risk of "rare but catastrophic" failures.

Pros: minimal space.
Cons: not credible for high-assurance use without strong evidence.

8.2.3 Recommended v1 redundancy approach (suggestion)

For v1, a conservative and credible default is:

- **One "Secret QR"** at ECC H
- **One "Backup Secret QR"** (identical payload) in a second location (back of the recovery note)
- **Human-readable fallback text** with strong checksum

8.2.4 How redundancy prevents "single character loss = dead funds"

- QR ECC allows recovery from damaged modules up to a threshold.
- A second QR means damage at one location doesn't kill recoverability.
- Human-readable fallback allows recovery even if both QRs fail, as long as the checksum validates.
- Segment-based redundancy can tolerate partial destruction beyond QR ECC limits.

In other words: redundancy ensures recoverability is not dependent on one perfect print surviving in one perfect spot.

8.3 Tamper evidence (baseline v1)

8.3.1 Design intent (v1 scope)

Tamper evidence in v1 aims to provide:

- **obscuration** of the secret area before intended opening, and
- a clear **irreversible indication** that the secret area was accessed and therefore that the funds in the note have been redeemed or converted (the note has been used).

v1 intended security claim (limited):

- "The secret area is obscured before opening and opening creates visible evidence."

It does **not** claim that copying is impossible or that a determined adversary cannot extract the secret.

8.3.2 Sticker mechanism (recommended design features)

A credible v1 tamper layer typically includes:

1. **Obfuscation background layer**

- A printed obfuscation pattern that prevents reading QR/text through the sticker under normal lighting and angle.

2. **Tamper-evident adhesive / VOID pattern**

- When lifted or opened, leaves:
    - a “VOID/OPENED” message,
    - delamination,
    - residue pattern,
    - or a split layer that cannot be reassembled cleanly.

3. **Edge/border warning**

- Border text that becomes visibly broken or misaligned if lifted.

4. **Alignment markers**

- Visual reference marks to spot substitution or resealing.

8.3.3 Placement and interaction constraints (important)

The tamper sticker must not undermine the primary objective: recoverability.

Mandatory constraints:

- Sticker adhesive must not chemically interact with ink/substrate in a way that degrades QR/text over time.
- The sticker must not cause ghosting, bleed-through, or migration that lowers contrast beyond scan threshold.
- Opening the sticker must not shred the QR/text beyond recoverability if the design expects the user to open it during recovery.

8.3.4 Verification procedure (must be documented)

Before opening, user should be able to verify:

- sticker is intact,
- no edge lift,
- alignment marks match,
- no residue or bubble patterns suggesting reseal.

After opening, user should see:

- expected VOID/opened evidence,

- no unusual patterns that suggest prior opening.

8.3.5 Testing requirements (links to Chapter 6)

- Baseline peel/lift attempts
- Aging tests (humidity/heat)
- Adhesive compatibility tests
- User inspection usability test (“can normal users tell intact vs opened?”)

8.3.6 Implementation placeholders (must be resolved before production)

- Payload content and encoding spec: **[TBD]**
- QR module size and ECC validated via device-matrix testing: **[TBD]**
- Fallback encoding (Base32/Base58) and checksum scheme: **[TBD]**
- Redundancy selection: **[Option 1/2/3/4]**
- Sticker material spec and supplier: **[TBD]**

9 RELIABILITY & SURVIVABILITY TESTING (PRE-LAUNCH GATE)

This chapter defines the **pre-launch reliability qualification** program for Stablenotes v1 recovery media and associated tamper layers. In v1, survivability is not a quality preference—it is a **fund-loss risk control**. An unreadable key is functionally equivalent to stolen funds.

The program is designed to be (i) testable, (ii) repeatable across production lots, and (iii) defensible to integration partners and auditors.

9.1 Service-life targets

Stablenotes v1 must define two separate service-life targets: **carry-life** (exposed to handling) and **storage-life** (stored offline with limited handling). These must be explicit to avoid overclaiming.

9.1.1 Carry-life target (suggested)

Definition: The period during which a note may be carried, handled, or accessed periodically (e.g., kept in a wallet, bag, or safe access location).

- **Carry-life target:** 24 months
- **Recommended initial target: 12 months** carry-life under defined handling assumptions.

Assumptions (must be stated):

- moderate daily handling,
- occasional folding,
- occasional exposure to moisture/sweat,
- typical indoor lighting and temperature.

9.1.2 Storage-life target (suggested)

Definition: The period during which a note is stored offline with minimal handling (e.g., stored at home, in a safe, or protected sleeve).

- **Storage-life target:** 20 years
- **Recommended initial target: 5–10 years** storage-life (v1), with any longer target treated as future validated claim.

Assumptions (must be stated in marketing material):

- stored in protective sleeve,
- low UV exposure,
- stable humidity/temperature,
- no chemical contact.

9.2 Test matrix

The test program must cover both **baseline qualification** (design validation) and **ongoing QC sampling** (lot acceptance). It must also cover the complete media stack: ink + substrate + overlays (security sticker).

The readability and overall quality of a QR code are governed by international standards: ISO/IEC 15415 (for printed labels) and ISO/IEC 29158 (for Direct Part Marking or DPM) These standards will be followed in a very strict manner using the best possible testing equipment and verification software available on the market.

9.2.1 Key Parameters for QR Code Readability

Standard verification software assesses these parameters to assign an overall grade (usually A through F or 4.0 through 0.0):

- **Symbol Contrast (SC):** Measures the difference between the highest and lowest reflectance values. High contrast between black modules and the white background is essential.
- **Modulation (MOD):** Evaluates the uniformity of reflectance across the code. Low modulation occurs when some "dark" modules are too light or "light" modules are too dark, confusing the scanner.
- **Reflectance Margin (RM):** Determines how well each module can be distinguished as light or dark relative to a global threshold.
- **Fixed Pattern Damage (FPD):** Assesses the integrity of the "Finder Patterns" (the three large squares in the corners) and "Alignment Patterns." If these are damaged, the scanner cannot "locate" the code.
- **Axial Non-uniformity (ANU):** Measures if the code is stretched or scaled differently on one axis versus the other.
- **Grid Non-uniformity (GNU):** Evaluates how well the modules align with an ideal grid. Significant deviation suggests printing distortion.
- **Unused Error Correction (UEC):** QR codes have built-in redundancy (Reed-Solomon). This parameter measures how much of that "safety net" is still available. If this is low, the code is "barely" readable.

9.2.2 Mechanical tests (mandatory)

Purpose: simulate daily handling and physical wear that can destroy QR modules or characters.

Minimum tests:

- **Abrasion / rub tests**
  - dry rub cycles with controlled pressure
  - wet rub cycles (light moisture)
- **Fold cycles**
  - repeated folding along defined fold lines
  - crease stress with compression

- **Edge wear**
  - repeated insertion/removal from sleeve or wallet pocket
- **Smudge resistance**
  - finger rub and cloth rub tests

9.2.3 Water and humidity tests (mandatory)

Purpose: determine whether accidental exposure causes unrecoverable damage.

Minimum tests:

- **Splash exposure**
  - water droplets + wipe
- **Immersion (short duration)**
  - e.g., 5–30 seconds immersion, followed by air-dry
- **Humidity cycling**
  - high humidity exposure (e.g., 85% RH) and return to ambient
- **Condensation simulation**
  - cold-to-warm transition

9.2.4 Chemical exposure tests (mandatory)

Purpose: reflect common real-world contamination that destroys readability.

Minimum substances:

- **Sweat / skin oil simulation**
- **Hand sanitizer** (alcohol-based)
- **Detergent/soap residue**
- **Beverages:** water, coffee/tea, soda (sugar + acid exposure)
- **Household cleaner** (mild) — optional depending on target market

Procedure concept:

- controlled application → dwell time → wipe/dry → scan and legibility test.

9.2.5 Aging tests (mandatory)

Purpose: accelerated aging to estimate service life under storage conditions.

Minimum tests:

- **UV exposure**
  - defined spectrum and dose
- **Heat aging**
  - elevated temperature exposure (e.g., 60–80°C for defined hours)
- **Cold cycling**
  - low temperature + return to ambient

- **Thermal cycling**
    - repeated hot/cold transitions

9.2.6 Sticker aging and adhesive interaction (mandatory if sticker used)

Purpose: ensure the tamper layer does not degrade recoverability.

Minimum tests:

- **Adhesive migration**
    - observe bleed-through, residue, ghosting over time
- **Edge lifting**
    - humidity/heat to test peel behavior
- **Residue and delamination behavior**
    - post-aging opening test

9.3 Acceptance criteria

Acceptance criteria must be defined to support:

- binary ship gate decisions,
- partner diligence,
- ongoing QC acceptance,
- and incident root-cause analysis.

9.3.1 Scan success thresholds (QR)

To get a formal assessment beyond a simple "it scans on my phone," industries use Bar Code Verifiers. These devices use calibrated lighting and high-resolution optics. The use of such device is mandatory during testing and fine-tuning. On top of that, testing should be pragmatic and therefore must also include representative:

- low/mid/high Android devices
- iPhone models
- in typical indoor and low-light conditions

Core metrics (mandatory):

- **Scan success rate:** ≥ **98.5%** after each stress test
    - **Recommended initial threshold: ≥ 95%** scan success across device matrix post-stress
- **Time-to-scan:** ≤ **2 seconds** (median) and ≤ **5 seconds** (95th percentile)
    - Suggested starting point: median ≤ 5s; p95 ≤ 7s (to be validated in lab)
- **Failure classification:** every failure must be categorized (contrast, blur, distortion, quiet zone damage, camera limitations).

Hard stop criteria (suggested):

- Any single stress category causing scan success < **90.0%** triggers design review.
- Any repeatable failure mode across lots triggers batch hold and investigation.

9.3.2 Human-readable fallback legibility thresholds

Fallback text must remain readable when QR fails.

Core metrics:

- **Legibility success:** ≥ **99.9%**
  - Recommended initial: **≥ 99.5%** legibility in trained-operator reading tests post-stress.
- **Checksum validation:** 100% of transcribed fallbacks must pass checksum where legible.
- **Ambiguity rate:** misread characters must be below a defined threshold (e.g., ≤ 0.1%) by design (font/encoding selection).

9.3.3 Sample sizes and QA acceptance rules

Two layers:

A) Design validation (pre-launch qualification)

- **Sample size:** 200 notes per media configuration per test condition
  - Suggested minimum: **n = 100** per condition for QR pass-rate confidence, higher for critical conditions.
- Test across at least 3 production lots to ensure repeatability.
- Independent lab required for baseline qualification report.

B) Ongoing production QC (lot acceptance)

- Each manufactured lot must have:
  - sampling plan (AQL-based or risk-based)
  - print contrast check
  - Serial QR scan check
  - fallback checksum verification
- **Lot acceptance:** lot passes only if all sampled notes meet thresholds; otherwise, lot held or reworked per SOP.

9.4 Replacement and support policy

This policy is part of the safety case. It defines what happens when survivability fails in practice.

9.4.1 User-facing support outcomes (required)

When users report damage/unreadable media, the support process must route into one of these outcomes:

1. **Recoverable note with degraded QR**

- If fallback text is readable and checksum-valid:
    - guide user through recovery using fallback input.
    - encourage immediate migration of funds and re-issuance.

2. **Partially readable note**

- Provide structured guidance:
    - attempt scan under varied lighting / camera settings
    - attempt fallback using partial blocks + checksum logic (if supported)
    - if not recoverable: communicate clearly that self-custody loss may be irreversible.

3. **Unreadable note**

- If both QR and fallback are unreadable:
    - classify as potential unrecoverable loss event.
    - log incident with batch/serial details for investigation.
    - provide disclaimers and best-effort remediation steps (but no false guarantees).

9.4.2 Replacement policy (operational)

Stablenotes should define a replacement policy by product class:

- **Blank recovery media (Class N):**
    - replacement available for defective media within warranty window, subject to verification.
- **Funded note wallets (user value at risk):**
    - replacement of media does not restore funds; only recovery does.
    - support focuses on enabling recovery and immediate migration.

9.4.3 Disclaimers (required; must align with legal posture)

- Printed recovery media is a self-custody backup tool.
- Damage, loss, or exposure may cause permanent loss of funds.
- Users are responsible for safe storage and handling.
- Survivability targets assume defined environmental conditions.

9.4.4 Recommended user best practices (optional but strongly recommended)

Provide clear guidance:

- Store notes in **protective sleeves** (provided or recommended spec).
- Avoid moisture, direct sunlight, and chemicals.
- Avoid repeated folding; store flat when possible.
- Verify readability periodically (optional; guidance to avoid exposing secrets).

- If tamper layer is opened, **recover promptly and re-issue**; do not re-store opened notes.

9.4.5 Operational feedback loop (required)

Every survivability incident should:

- record note serial + batch (if available),
- capture failure mode and photos (user opt-in),
- map to lot/material traceability,
- feed into corrective action (CAPA) process:
    - lot hold/quarantine if systemic
    - supplier/material change if recurring
    - design updates if needed.

9.4.6 Implementation placeholders to finalize for production

- Carry-life duration: **[TBD]**
- Storage-life duration: **[TBD]**
- Test protocol references (lab + internal): **[TBD]**
- Device scanning matrix list: **[TBD]**
- Final acceptance thresholds (QR pass rate, time-to-scan, legibility): **[TBD]**
- Sampling plan (n, lots, AQL): **[TBD]**
- Warranty/replacement terms and support scripts: **[TBD]**

10 SECURITY TESTING & ASSURANCE (PRE-LAUNCH VS FUTURE)

This chapter defines the **security assurance program** for Stablenotes. It separates what is **mandatory before any v1 production deployment** from what is **explicitly future-gated** and out of scope for v1. The goal is to prevent "security theater": every claim must map to an evidence artifact, and every future claim must be gated behind specific tests and pass criteria.

Scope note: Stablenotes v1 is a recovery/cold-storage layer. It is not a bearer-transfer product. Therefore, v1 assurance focuses on (i) key generation correctness, (ii) key non-retention, (iii) firmware/update integrity, (iv) companion app safety, and (v) supply-chain/media survivability. Any v2 bearer-transfer assurance is **not shipped** and must not be marketed or implied.

10.1 Pre-launch mandatory audits (v1 ship gates)

Stablenotes v1 MUST complete independent reviews covering:
A) key generation and non-retention, B) device firmware and update chain, and C) companion app and SDK integration safety.

10.1.1 Key generation and non-retention audit (mandatory)

Purpose: Prevent catastrophic loss from weak randomness, key leakage, or secret retention.

Scope (minimum):

1. **RNG correctness**
    - Confirm exclusive use of platform CSPRNG sources (no custom RNG, no manual seeding).
    - Verify no insecure fallback RNG providers.
    - Verify uniqueness controls (no repeated keys).
2. **Key lifecycle and memory hygiene**
    - Identify all buffers and objects that may contain secrets.
    - Review secret handling for copies, string conversions, serialization, caching.
    - Verify zeroization/invalidation procedures (best-effort per platform).
3. **Non-retention proof**
    - Confirm secrets are not written to persistent storage (files, databases, preferences).
    - Confirm secrets are not present in logs, analytics, crash reports, or telemetry.
    - Confirm secrets are not transmitted off-device.
4. **Crash/interrupt paths**

- Validate behavior under app kill, device restart, backgrounding, low-memory.
- Confirm sensitive UI redaction where applicable.
- Confirm no residual secret material appears in crash artifacts.

5. **Print-flow verification gate**
   - Confirm “no irreversible step without verification” is enforced (scan + checksum).
   - Confirm any failures lead to safe abort and do not create “half-issued” dangerous states.

Deliverables (evidence artifacts):

- Keygen_RNG_Independent_Audit.pdf
- Non-Retention_Runtime_Test_Report.pdf
- Crash_Interrupt_Path_Review.pdf
- Sensitive_Flow_Verification_Gate_Test_Report.pdf

Acceptance criteria (ship gate):

- No Critical/High findings open at launch.
- No evidence of secret persistence, logging, or off-device leakage in tested builds.
- Verified absence of manual seeding and insecure RNG fallbacks.

10.1.2 Firmware and update chain audit (mandatory)

Purpose: Ensure the printer unit cannot be subverted via firmware compromise, downgrade attacks, or supply-chain injection.

Scope (minimum):

1. **Secure boot and chain-of-trust**
   - Verify secure boot implementation, signature verification, and enforcement.
   - Verify debug interfaces locked/disabled in production.
2. **Firmware attack surface review**
   - Review connectivity interfaces (USB/BLE/Wi-Fi) and privileged commands.
   - Verify authentication/authorization for maintenance modes.
   - Confirm sensitive payloads are not logged/persisted.
3. **Update security**
   - Review signed update flow: verification-before-install.
   - Validate key hierarchy and signing key custody (HSM policy, role separation).

- Validate anti-rollback controls.
- Validate staged rollout + emergency pause procedures.

4. **Supply chain / provisioning controls (high level)**
   - Verify only signed firmware can be loaded at factory.
   - Verify device identity provisioning controls and traceability.

Deliverables (evidence artifacts):

- Firmware_Secure_Boot_Update_Chain_Audit.pdf
- Anti_Rollback_Downgrade_Attack_TestReport.pdf
- Provisioning_Security_Assurance_Summary.pdf *(relevant for v2 only)*
- Debug_Interface_Lockdown_Verification_Report.pdf

Acceptance criteria (ship gate):

- Secure boot enforced; unsigned firmware cannot run.
- Downgrade attacks fail; anti-rollback is effective.
- Update signing process has documented key custody and two-person integrity for release.
- Factory provisioning process prevents firmware side-loading.

10.1.3 Companion app review (mandatory)

Purpose: Reduce the risk that the "untrusted phone" leaks secrets, misleads users, or compromises the v1 security posture.

Scope (minimum):

1. **Scope enforcement**
   - Verify v1 mode: verification/recovery only.
   - Confirm prohibited v1 features are not enabled (no custody, no "bank-like" claims).

2. **Third-party SDK inventory and risk**
   - Identify analytics/crash/reporting SDKs and disable session replay on sensitive screens.
   - Confirm privacy disclosures match reality.

3. **Transaction safety and UI integrity**
   - Verify warnings for irreversible actions and secret-handling steps.
   - Confirm safe error handling and retry behavior.

4. **Data flow boundary review**
   - Confirm no KYC artifacts or identity data handled by the app when third-party ramps are integrated (if applicable).

Deliverables (evidence artifacts):

- App_Security_Review.pdf
- Third-PartySDK_Inventory_and_Risk_Review.pdf
- Sensitive_Screen_No-Replay_Redaction_Evidence.pdf
- Data_Flow_Privacy_Boundary_Assessment.pdf

Acceptance criteria (ship gate):

- No Critical/High findings open.
- No session replay or sensitive telemetry on key-handling screens.
- No contradictions between app-store privacy declarations and actual data handling (if applicable).

10.2 Future work (v2) explicitly gated (NOT shipped in v1)

This section exists to prevent accidental scope creep and over-claiming. The following tests are **v2-only** and are not required for v1 shipping but MUST be passed before any product is marketed or deployed as bearer-transfer/offline transactional.

10.2.1 Required v2 adversarial-transfer tests (examples)

1. **Copy-without-open**
    - Attempt to read/photograph/extract secret through sticker without visible tamper evidence.
    - Test across lighting, angles, optics, and common enhancement techniques.
2. **Open-and-reseal**
    - Attempt to peel, open, copy, and reseal such that normal inspection cannot detect prior opening.
3. **Swap attack**
    - Replace a genuine sticker with a counterfeit or swapped sticker/overlay.
    - Replace genuine note with a counterfeit note carrying attacker-controlled secrets.
4. **Counterfeit replication**
    - Produce counterfeit notes/stickers that pass lay inspection and initial validation procedures.
5. **Field deception test**
    - Evaluate whether non-expert agents/users can reliably detect tamper/counterfeit using the provided SOP.

v2 gating statement (mandatory):

- These v2 tests are **not shipped in v1**.

- No bearer-transfer product may be launched, marketed, or piloted until:
    - independent adversarial testing confirms success rates below defined thresholds, and
    - field detection procedures meet acceptable false-accept/false-reject rates.

10.3 Disclosure & bounty policy

A security product must assume vulnerabilities will be found. The question is whether the organization can detect, respond, and remediate with discipline.

10.3.1 Vulnerability reporting policy (VDP)

Stablenotes MUST maintain a published vulnerability disclosure policy that includes:

- a dedicated security contact (email + PGP key or HackerOne)
- scope of systems covered (SDK, app, firmware, smart contracts, media workflows)
- safe-harbor language for good-faith research
- reporting format expectations (PoC, logs, affected versions)
- acknowledgment and response timelines

10.3.2 Bug bounty program (recommended; phased)

A bug bounty SHOULD be implemented in phases:

- **Phase 1 (private bounty):** invite vetted researchers prior to launch.
- **Phase 2 (public bounty):** expand scope post-launch once triage capacity is proven.

Bounty scope should explicitly include:

- key generation and non-retention failures,
- device update chain compromise,
- app telemetry leakage,
- smart contract vulnerabilities,
- supply-chain/media integrity issues (where researchers can test safely).

10.3.3 Patch and remediation SLAs (mandatory)

Define and publish internal SLAs for:

- critical vulnerabilities (e.g., 7–14 days)
- high severity (e.g., 30 days)
- medium/low (scheduled)

For firmware, define staged rollout and emergency pause procedures (see Chapter 7).

10.3.4 Post-launch monitoring commitments (mandatory)

At minimum, Stablenotes MUST maintain:

- release monitoring and incident logging
- anomaly detection for duplicate keys/public addresses (privacy-safe)
- monitoring of failure rates in printing verification and scan success (field telemetry allowlist)
- partner integration certification refresh checks (for SDK deployments)
- periodic security reviews for third-party SDK changes

10.3.5 Public communication policy (mandatory)

Security communications must be controlled and factual:

- publish security advisories when users/partners need to act
- avoid ambiguous claims or minimization
- provide clear remediation instructions and upgrade requirements

10.3.6 Implementation placeholders to be resolved prior to v1 production

- Selected auditors and scope sign-off: **[TBD]**
- Audit schedule and gating dates: **[TBD]**
- Bug bounty platform and rules: **[TBD]**
- Severity scoring model (CVSS + product-specific impact): **[TBD]**
- Monitoring telemetry allowlist and retention: **[TBD]**
- Partner certification refresh cadence: **[TBD]**

11 REGULATORY POSITIONING (EU-PRIMARY)

This chapter is intentionally short and factual. It should be read together with the Whitepaper Legal Disclaimer, which governs and qualifies all statements in this section.

11.1 What v1 is not

Stablenotes v1 is positioned as a **non-custodial, self-custody recovery layer** delivered as software (SDK + reference implementation) and associated recovery media specifications. Under this v1 scope, the Technology is **not**:

- **Not custody:** PrintDreams does not take custody of, control, access, or manage user crypto-assets or private keys. Users retain exclusive control of keys and recovery materials (subject to user handling and device security assumptions).
- **Not a payment service:** v1 is not a payment instrument, payment initiation service, money remittance service, or e-money product under EU payments frameworks. It does not operate a merchant acceptance network, cash-out network, or any operator/agent program.
- **Not an exchange / on-ramp / off-ramp:** v1 does not provide exchange execution, brokerage, or fiat on/off-ramp services. Where third-party services (e.g., regulated ramps) are integrated in a partner wallet, those services are provided by the third-party provider under its own terms and regulatory permissions (see Disclaimer).

Operational boundary (v1): the "print" workflow is framed and implemented as **backup/recovery for self-custody**, not as a bearer-transfer instrument or a cash replacement.

11.2 Jurisdictional note (EU focus)

At a high level, Stablenotes v1 is intended to align with the category of **non-custodial wallet software and supporting recovery tooling**, where the provider does not hold or control client crypto-assets or private keys.

As an EU reference point, MiCA includes recital language stating that **hardware or software providers of non-custodial wallets should not fall within the scope of the Regulation**. (EUR-Lex)

Important qualifier: regulatory classification is fact-dependent. If a deployment expands beyond the v1 boundary (e.g., custody, execution of orders, exchange, or other crypto-asset services performed "professionally"), MiCA CASP obligations and other EU regimes may become relevant. The v1 strategy therefore relies on strict scope control and clear separation of third-party regulated services (as reflected in the Disclaimer and partner integration controls).

12 ROADMAP & GATES

This chapter sets out a two-track roadmap with explicit **Go/No-Go** gates and **kill criteria**. The intent is to (i) keep Stablenotes v1 deployable and investable as an SDK-first recovery layer, and (ii) prevent v2 ambitions from contaminating v1 scope, compliance posture, or security claims.

Rule: Track A (v1) ships only on evidence. Track B (v2) proceeds only if Track A proves viability and Track B gates are passed.

12.1 Track A (v1): SDK-First Recovery Layer (Production Path)

12.1.1 Milestones and required evidence artifacts

Below is the recommended milestone sequence. Each milestone is paired with required evidence artifacts. All "TBD" fields must be completed with owners and dates prior to execution.

Milestone A0 — Scope lock and partner posture (TBD: Date)

Objective: Freeze v1 scope as recovery-only and ensure partner-facing posture is consistent.

Required artifacts:

- V1_Scope_Definition.pdf (explicit non-goals)
- Marketing_Claims_Policy_v1.pdf
- Partner_Integration_Boundary_Spec_v1.pdf

Gate: No v1 deployment proceeds if v1 copy/UX implies cash-like transfer or bearer usage.

Milestone A1 — Integration readiness (TBD: Date)

Objective: SDK and reference implementation support core lifecycle flows.

Required artifacts:

- SDK_API_Spec_v1.md + sample integration
- Partner_Integration_Certification_Checklist_v1.pdf
- Reference_Wallet_Release_Notes_<ver>.pdf *(placeholder)*

Gate: "Hello world" integration must be achievable in ≤ **[TBD]** engineering days using the "golden path."

Milestone A2 — Security baseline audits complete (TBD: Date)

Objective: Prove v1 security fundamentals: RNG, non-retention, firmware/update integrity.

Required artifacts (minimum):

- Keygen_RNG_Independent_Audit.pdf

- Non-Retention_Runtime_Test_Report.pdf
- Firmware_Secure_Boot_Update_Chain_Audit.pdf
- Anti-Rollback_Downgrade_Attack_Test_Report.pdf
- Third-Party_SDK_Inventory_and_Risk_Review.pdf

Gate: No open **Critical/High** findings; remediation verified.

Milestone A3 — Durability qualification complete (TBD: Date)

Objective: Prove printed recovery media survivability under defined service-life targets.

Required artifacts (minimum):

- Durability_Test_Protocol_v1.pdf
- Independent_Durability_Lab_Report.pdf
- Sticker_Adhesive_Compatibility_Report.pdf
- Supported_Device_Matrix_v1.pdf

Gate: QR scan success and fallback legibility meet thresholds in Chapter 9/Appendix

---

Milestone A4 — Supply chain integrity and traceability operational (TBD: Date)

Objective: Establish banknote-like traceability controls for media production and distribution.

Required artifacts:

- Serialized_Note_Spec_v1.pdf
- Batch_Manifest_Format_v1.pdf
- ChainOfCustody_SOP_v1.pdf
- Quarantine_Recall_SOP_OnePager.pdf
- Quarantine drill: Quarantine_Drill_Report_<date>.pdf

Gate: Quarantine-by-serial-range drill meets SLA: quarantine initiated ≤ **[TBD]** hours and propagated ≤ **[TBD]** hours.

Milestone A5 — Controlled production pilot with design partner (TBD: Date)

Objective: Deploy v1 with a design partner under caps to generate field evidence.

Required artifacts:

- Pilot_Spec_<partner>_v1.pdf (caps, geos, assets, support model)
- Incident_Response_Runbook_Joint_v1.pdf
- Pilot_Metrics_Dashboard_<period>.pdf

Gate: Pilot KPIs meet thresholds (below) with no critical incidents.

Milestone A6 — Production launch (TBD: Date)

Objective: Scale to production partner deployment(s) using the certified integration program.

Required artifacts:

- Evidence_Pack_Index_v1.pdf (claims-to-evidence mapping)
- Release_Governance_and_Change_Control_SOP.pdf
- Partner_Production_Readiness_Sign-off_<partner>.pdf

Gate: All ship-gate controls are GREEN; partner operations trained; ongoing monitoring operational.

12.1.2 Go/No-Go conditions (v1)

A v1 deployment is **GO** only if all are true:

Security gates (must-pass):

- Independent audits confirm RNG correctness and non-retention (Chapter 10).
- Secure boot, signed updates, and rollback protection verified (Chapter 7).
- No unresolved critical/high issues.

Reliability gates (must-pass):

- Durability lab results meet scan success + legibility thresholds (Chapter 9).
- Post-print verification gate enforced.

Operational gates (must-pass):

- Supply chain traceability operational; quarantine drill passed (Chapter 6 supply chain).
- Partner integration certified and support runbooks in place.

No-Go triggers (immediate):

- Evidence of key leakage or secret retention in production builds.
- Systemic unreadable-media incidents above threshold.
- Firmware/update integrity compromise.
- Partner refuses required boundary controls or claims discipline.

12.1.3 v1 kill criteria (hard)

If any of the following occurs, Track A should be paused or terminated unless remediated with verified evidence:

1. **Catastrophic key compromise** (RNG failure, non-retention breach, firmware compromise) with credible exploit path.
2. **Unrecoverable loss rate** exceeding **[TBD threshold]** under normal use (unreadable QR/fallback without viable recovery).
3. **Support burden** indicates the product is unsafe for non-expert users at scale (e.g., critical user error rate above **[TBD]**).

4. **Partner adoption stalls**: no additional partners move from LOI → integration within **[TBD 6–9 months]** despite evidence pack completion.

12.2 Track B (v2): Separate Program (Future-Gated)

Track B is a separate program that explores bearer-transfer/offline transactional use cases. It is **not part of v1** and must not be implied by v1 claims, marketing, or partner deployments.

12.2.1 Entry criteria (v2 cannot start until these are met)

Track B may proceed only if:

1. **Track A proves viability**
    - at least **[TBD: 1–2]** production partner deployments OR equivalent traction signal
    - durability and incident rates within target thresholds for **[TBD: 3–6 months]**
2. **Security maturity is proven**
    - v1 evidence pack is complete and maintained
    - security operations (VDP/bounty/IR) functioning in practice
3. **Regulatory and liability pathway exists**
    - credible sponsor/authorization strategy in the target pilot geo(s) *(placeholder)*
    - preliminary liability/reserve framework is economically viable *(placeholder)*

12.2.2 v2 kill criteria (hard)

Track B must be killed (or paused indefinitely) if any occur:

1. **Adversarial transfer cannot be secured**
    - independent red-team testing shows practical copy-without-open or reseal success rates above acceptable thresholds.
2. **Field detectability fails**
    - non-expert detection SOP produces unacceptable false-accept rates (tampered/counterfeit notes pass inspection).
3. **Regulatory pathway is not credible**
    - no viable sponsor/authorization path within **[TBD timeframe]** despite sustained effort, or requirements make the business uneconomic.
4. **Loss/fraud economics are untenable**
    - required reserves/insurance or expected loss rates make unit economics non-viable.
5. **v2 contaminates v1**

- if v2 work causes v1 to drift into “cash replacement” positioning, increases compliance exposure, or jeopardizes key partners.

12.2.3 Placeholders to finalize

- Target dates per milestone: **[TBD]**
- Partner names and pilot caps: **[TBD]**
- Audit firms and lab vendors: **[TBD]**
- Quantitative thresholds (unrecoverable loss rate, scan success, support burden): **[TBD]**
- v2 security thresholds and red-team rounds: **[TBD]**

APPENDICES (EVIDENCE-FIRST)

These appendices are designed to make the whitepaper **auditable**. Anything that is a security/reliability claim in the main body should map to an artifact here (or to the data room), not remain a narrative assertion.

Appendix A — Claims-to-Evidence Matrix (master index)

Purpose: Single source of truth linking each claim to: owner, acceptance criteria, and evidence artifact(s).
Artifact: Claims_to_Evidence_Matrix_Template.xlsx *(maintained as a living document; versioned in the data room).*
Contents (minimum fields): Claim ID, claim statement, scope (v1/v2), risk class, evidence required, owner, acceptance criteria, artifact link, status.

Note: v2 claims must remain clearly tagged as "future-gated" and must not be used for v1 marketing or partner commitments.

Appendix B — Smart Contract ABI / Message Interfaces (per chain)

Purpose: Provide chain-specific interfaces that correspond to Chapter 5.

B.1 Deployment summary (per chain)

- Chain: **[TON / Other TBD]**
- Contract name: **[RecoveryPolicyContract / NoteRegistryContract / TokenVerifier]**
- Address: **TBD at deployment**
- Version: **[vX.Y.Z]**
- Upgradeability: **[Immutable / Upgradeable with policy]**
- Audit reference: **[Audit report ID/date]**

B.2 ABI / message schema

Provide one of the following formats per chain:

- **TON:** message handlers + TL-B schema / opcodes and payload definitions *(placeholder)*
- **EVM chains (if later supported):** JSON ABI files per contract *(placeholder)*

Required inclusions:

- Function/message definitions
- Event definitions (or equivalent)
- Error codes / revert reasons (or equivalent)
- State variables / getters (where applicable)

Artifact examples:

- TON_RecoveryPolicy_TLB_Schema_v1.txt
- TON_RecoveryPolicy_Opcode_Map_v1.md
- EVM_RecoveryPolicy_ABI_v1.json *(if applicable)*

Appendix C — Test Protocols (Security + Reliability)

Purpose: The assurance program must be reproducible by independent labs/auditors.

C.1 Security testing protocols

Include:

- RNG/key generation validation protocol
- Non-retention / leakage testing protocol (logs, storage, crash dumps)
- Crash/interrupt-path testing protocol
- Firmware secure boot verification protocol
- Update signing + anti-rollback test protocol
- Third-party SDK telemetry/session replay audit protocol

Artifact examples:

- Protocol_RNG_Keygen_v1.pdf
- Protocol_NonRetention_RuntimeLeakage_v1.pdf
- Protocol_CrashInterrupt_v1.pdf
- Protocol_SecureBoot_v1.pdf
- Protocol_UpdateChain_AntiRollback_v1.pdf
- Protocol_ThirdPartySDK_Audit_v1.pdf

C.2 Reliability and survivability protocols

Include:

- Media durability test matrix (mechanical, water/humidity, chemical, UV/thermal)
- Scan-device matrix protocol (devices, lighting, pass/fail)
- Sticker/adhesive compatibility protocol
- Post-print verification gate protocol
- Lot acceptance sampling plan (AQL or risk-based)

Artifact examples:

- Protocol_Durability_TestMatrix_v1.pdf
- Protocol_DeviceScanMatrix_v1.pdf
- Protocol_AdhesiveMigration_v1.pdf
- Protocol_PostPrint_VerificationGate_v1.pdf
- Protocol_LotAcceptance_Sampling_v1.pdf

Evidence-first rule: Protocols must specify parameters, sample sizes, acceptance criteria, and reporting format.

# # #