

RFC-002: LanternSeaCoin Magnetic Stripe Voucher Extension

Status: Draft

Authors: LanternSeaCoin Protocol Team

Date: 6 October 2026

Language: English

Depends on: RFC-001 (NFC Extension), LanternSeaCoin Reference Paper 2.0 (§5, §14.1, §18, §19, §20)

Scope: Single-use LSC vouchers carried on magnetic stripe cards, online redemption, issuance register, clone handling

1. Abstract

This RFC specifies how a LanternSeaCoin (LSC) voucher can be carried on a magnetic stripe card and redeemed at a terminal.

A magnetic stripe is static and can be copied completely. The design therefore does not try to make the carrier secure. Instead it relies on four properties:

1. the stripe carries a 128-bit voucher secret, and the issuer register stores only a commitment to it;
2. a voucher is redeemed exactly once and only online;
3. the face value is limited by the loss the issuer is willing to accept from a copied stripe;
4. every second redemption attempt is detected and treated as clone evidence.

A voucher is backed by an LSC stamp (Reference Paper §5) and settles through the online redeem flow of RFC-001 §8. This RFC does not define payment cards, and a voucher is not a payment card.

2. Motivation

Magnetic stripe readers are widespread, cheap and accepted by many terminals. They can carry LSC value to places where no NFC device, chip card or smartphone is available.

A stripe has no secure element, no counter and no way to hide data. Copying it is trivial. A specification that pretends otherwise would be wrong. This RFC states the limits openly and bounds the damage:

- the loss from one copied voucher is at most its face value;
 - the loss from a batch of n vouchers is at most n times the face-value limit;
 - a database leak at the issuer does not allow redemption, because the register stores commitments, not secrets.
-

3. Terminology

Term	Meaning
Voucher	A single-use claim on a fixed LSC amount, represented by a secret printed or encoded on a stripe
Voucher secret k	128-bit random value carried on the stripe
Commitment C	$H(\text{enc}(\text{DOM_CL}, k, \text{issuer_id}))$, stored in the issuer register
Issuer register	Issuer-operated table of commitments with state, face value and validity
Backing stamp	The LSC stamp (non-movable coin) that funds the voucher

Term	Meaning
Redemption service	Issuer-operated online service that verifies and settles redemptions
Terminal	Certified or registered device with a magnetic stripe reader that submits redemption requests
Clone	A second physical or logical copy of the same stripe data
p_{clone}	Probability that a stripe is copied before redemption; fixed at 1 in this RFC
ϵ_M	Maximum expected loss per voucher that the issuer accepts
H, enc	BLAKE2b-256 and the length-prefixed encoding of Reference Paper §14.1.2

4. Goals

This RFC aims to provide:

- a compact stripe record that fits common tracks;
- an issuer register that is safe against database leaks;
- single-use online redemption with idempotent retries;
- authenticated terminals;
- an explicit, formula-based value limit;
- detection of cloned stripes;
- compatibility with the RFC-001 stamp and online redeem model.

5. Non-goals

This RFC does not define:

- a payment card, a PAN layout or any use of payment networks;
- offline redemption of magnetic stripe vouchers;
- protection against copying of the stripe itself;
- a chip or NFC carrier (see RFC-001 and RFC-003);
- a consumer wallet or user interface standard;
- partial redemption or change (see Open Decisions).

6. Voucher Model

6.1 Backing

Every voucher is backed by a stamp S held by the issuer. The stamp is a non-movable coin (Reference Paper §5). Redemption executes the claim of S and a transfer to the redeemer address atomically, as in RFC-001 §8.3. The claim window of S MUST end no earlier than the voucher's `valid_until`.

6.2 Secret and commitment

At issuance the issuer generates k with a cryptographically secure random generator (128 bits). The register stores:

$$C = H(\text{enc}(\text{DOM_CL}, k, \text{issuer_id})) \quad \text{DOM_CL} = \text{"LSC-CLAIMSECRET-v1"}$$

The register MUST NOT store k . Knowing C does not allow redemption. Guessing a valid k succeeds after q attempts with probability at most $q \cdot 2^{-128}$.

6.3 Voucher record

Field	Size	Meaning
C	32 bytes	commitment (register key)
face_value	uint64	amount in base units (10^8 per LSC)
stamp_id	8 bytes	backing stamp
valid_until	uint64	chain height after which the voucher expires
state	enum	see §11
pin_tag	optional	PIN verifier (§8.3)
pin_tries	uint8	remaining PIN attempts

7. Stripe Record Encoding

7.1 Carrier

The record is written in ISO/IEC 7811 style on one track. Typical capacities are 79 alphanumeric characters on track 1, 40 numeric characters on track 2 and 107 numeric characters on track 3. The record in §7.2 uses track 1; the record in §7.3 uses track 3. Track 2 is too small for a 128-bit secret in decimal form and **MUST NOT** be used.

The record **MUST NOT** follow a payment card PAN layout. The format code after the start sentinel **MUST NOT** be the code used for bank cards. Readers and software **MUST** treat the data as an LSC voucher, never as a card number.

7.2 Track 1 record (alphanumeric)

% F LSC1 ^ N ^ K ^ X ? LRC

Element	Size	Content
%	1	start sentinel
F	1	format code reserved for individual card issuers (see Open Decisions)
LSC1	4	protocol tag and version
^	1	field separator
N	1	network code: m mainnet, t testnet
K	26	k in Base32 (RFC 4648 alphabet, no padding); the 2 padding bits MUST be zero
X	4	check characters (§7.4)
?	1	end sentinel
LRC	1	longitudinal redundancy check, as defined by the track format

The record has 41 characters plus the LRC and fits in 79 characters.

7.3 Track 3 record (numeric)

; V N K = X ? LRC

V is one digit (version, 1), N one digit (1 mainnet, 2 testnet), K is k as a zero-padded 39-digit decimal integer (2^{128} is smaller than 10^{39}), X is four check digits. The record has 48 characters and fits in 107.

7.4 Check characters

$\text{chk} = \text{first 20 bits of } H(\text{enc}(\text{"LSC-MSV-CHK-v1"}, \text{version}, \text{net}, k))$

encoded as 4 Base32 characters (track 1) or as $\text{chk} \bmod 10^4$ in 4 digits (track 3). The check detects misreads. It is not a security mechanism.

7.5 Reader requirements

The reader **MUST** deliver raw track data. A reader that encrypts or masks track data in payment-card mode **MUST** be configured for raw output, or a non-payment reader **MUST** be used. The voucher record is not payment card data (Reference Paper §19.1, R1 does not apply to it), but the terminal **MUST NOT** log k , and it **MUST** show only the last 4 characters of K on any receipt or display.

8. Issuance

8.1 Creation

1. Generate k ; compute C .
2. Create or reserve the backing stamp S for face_value .
3. Write the register entry with state ISSUED.
4. Encode and write the stripe (§7); verify it by reading it back.
5. Destroy k in the issuing system after the write.

8.2 Activation

A voucher is created inactive. It becomes ACTIVE through an activation call by an authorized point of sale. Redemption of an ISSUED voucher **MUST** fail with `E_VOUCHER_INACTIVE`. Activation limits the exposure of stolen, unsold stock.

8.3 Optional PIN

A voucher **MAY** have a PIN of 4 to 8 digits, issued separately from the stripe. The register stores a keyed verifier, not the PIN:

$\text{pin_tag} = \text{HMAC-SHA256}(K_{\text{pin}}, \text{enc}(C, \text{pin}))$ K_{pin} held in an HSM

PIN entry **MUST** take place on a PIN entry device, never on a general-purpose keypad. The service allows 3 attempts; after the third failure the state becomes LOCKED, and only the issuer can unlock the voucher.

8.4 Face-value limit

Because a stripe is fully copyable, $p_{\text{clone}} = 1$. The issuer **MUST** limit the face value as follows (value rule of Reference Paper §20.9):

$v_M \leq \varepsilon_M / p_{\text{clone}} = \varepsilon_M$ (no PIN)
 $v_M' \leq \varepsilon_M / (\text{tries} / 10^d)$ (PIN with d digits and `tries` attempts)

With $\varepsilon_M = 1$ LSC as reference value, a voucher without PIN carries at most 1 LSC. With a 4-digit PIN and 3 attempts the guess probability is $3 \cdot 10^{-4}$ and the limit rises to about 3,333 LSC; an issuer **SHOULD** use a much lower value. The loss from a batch of n vouchers is at most $n \cdot v_M$.

9. Redemption

9.1 Request

Redemption is online only. The terminal sends:

Field	Meaning
version, net	protocol version and network
terminal_id	registered terminal identity
ctr	terminal counter, strictly increasing
k	voucher secret read from the stripe
pin_block	optional encrypted PIN
payout_address	address that receives the amount (merchant or customer)
nonce	8 random bytes
terminal_sig	Ed25519 signature, tag LSC-MSV-REQUEST-v1, over all fields above

The request identifier is

```
request_id = H( enc( net, C, terminal_id, nonce ) )
```

The channel MUST use TLS 1.2 or later with authentication of the service. The terminal signature authenticates the request at application level.

9.2 Service procedure

The service MUST perform these steps in one durable transaction:

1. Verify terminal_sig; check that the terminal is registered and not revoked.
2. Check ctr against the highest counter seen for terminal_id.
3. Compute C from k; look up the register entry (E_VOUCHER_UNKNOWN if absent).
4. If request_id was seen: return the stored result for identical payloads; return E_REQUEST_REPLAY for a different payload.
5. Check state, validity and, if present, the PIN.
6. Set the state to REDEEMED and record the settlement intent (stamp S, payout address, face_value).

After the transaction commits, the service executes claim and transfer and retries until the transaction is included in a block. The voucher is consumed at commit. A failure of the payout never returns the voucher to ACTIVE; the issuer owes the payout.

9.3 Response

The service returns a status code (§13), the amount, a receipt reference and, once available, the transaction identifier. The response is final for the voucher but not for chain finality; the receipt MUST say so until the block is final (Reference Paper §16.5).

9.4 Idempotency

An identical request returns the identical result. A request with the same request_id and different content returns E_REQUEST_REPLAY. Terminals SHOULD retry after network failures with the same nonce.

10. Clone and Skimming Handling

A copy cannot be prevented. It is detected when a second redemption attempt arrives for a redeemed C. Then:

- the service returns `E_VOUCHER_CONSUMED`;
- it records `terminal_id`, time and request for both attempts;
- it raises a clone event for the issuer, who MAY revoke the remaining vouchers of the same batch or lot;
- the first committed redemption stays final.

Velocity limits SHOULD apply per terminal and per lot: a maximum number of redemptions per time window and a maximum total amount. Terminals that produce an unusual share of `E_VOUCHER_CONSUMED` results SHOULD be reviewed.

Exposure bound: for a lot of n vouchers with limit v_M , the maximum loss is $n \cdot v_M$.

11. State Machines

```
ISSUED --activate--> ACTIVE --redeem--> REDEEMED
ACTIVE --3 wrong PINs--> LOCKED --issuer unlock--> ACTIVE
ISSUED | ACTIVE | LOCKED --valid_until passed--> EXPIRED
ISSUED | ACTIVE | LOCKED --revoke--> REVOKED
```

REDEEMED, EXPIRED and REVOKED are final. An expired voucher's backing stamp returns to the issuer's control.

12. Consensus and Governance Rules

Settlement MUST obey the existing consensus rules. The register and the redemption service are issuer systems and are not part of consensus. Only the claim and the transfer are on-chain operations. This RFC does not authorize anyone to alter canonical state. Corrections follow the existing governance mechanism (Reference Paper §9).

13. Error Codes

Code	Meaning
<code>E_VERSION</code>	Unsupported version
<code>E_NETWORK</code>	Network mismatch
<code>E_FORMAT</code>	Stripe record malformed (sentinels, lengths, padding bits)
<code>E_CHECKSUM</code>	Check characters do not match
<code>E_SIGNATURE</code>	Terminal signature invalid
<code>E_TERMINAL_UNKNOWN</code>	Terminal not registered or revoked
<code>E_COUNTER</code>	Terminal counter not increasing
<code>E_VOUCHER_UNKNOWN</code>	No register entry for the commitment
<code>E_VOUCHER_INACTIVE</code>	Voucher not activated
<code>E_VOUCHER_CONSUMED</code>	Voucher already redeemed
<code>E_VOUCHER_EXPIRED</code>	Validity ended
<code>E_VOUCHER_REVOKED</code>	Voucher revoked
<code>E_VOUCHER_LOCKED</code>	Locked after wrong PINs

Code	Meaning
E_PIN_REQUIRED	PIN missing
E_PIN_WRONG	PIN incorrect; remaining attempts are returned
E_AMOUNT_LIMIT	Face value above the issuer limit
E_REQUEST_REPLAY	Request ID reused with a different payload
E_RATE_LIMIT	Velocity limit reached
E_TEMPORARY	Temporary service failure

14. Security Considerations

14.1 Copying and skimming

The stripe is fully copyable. This is accepted by design. The defences are the value limit (§8.4), activation (§8.2), optional PIN, validity limits, velocity limits and clone detection. A skimmer gains at most the face value per voucher, and only if it redeems before the owner.

14.2 Register leak

The register contains commitments only. An attacker who reads it cannot derive k , because the preimage is 128 bits of random data. The PIN verifier is keyed with an HSM key, so a leak does not allow offline PIN guessing.

14.3 Guessing

A guess hits a valid voucher with probability at most $q \cdot 2^{-128}$. The service MUST rate-limit unknown-voucher lookups per terminal and per source.

14.4 Terminal compromise

A compromised terminal learns k of vouchers it reads, which are anyway on the stripe. Terminal signatures and registration prevent unregistered devices from submitting. A revoked terminal's requests MUST fail.

14.5 Replay and man-in-the-middle

Requests carry a terminal counter, a nonce and a signature, and travel over TLS. A replayed request returns the stored result and creates no second payout.

14.6 Physical aspects

Demagnetization and wear cause misreads, which the check characters detect. They are an availability issue, not a security issue.

15. Privacy

The stripe carries no personal data. The voucher is a bearer instrument. The service sees terminal, time, amount and payout address. Operators MUST define retention and deletion of these logs under applicable data-protection law.

16. Compatibility

- The backing stamp, claim and transfer follow RFC-001 and Reference Paper §5, §12.3.
 - The value rule is the one of Reference Paper §20.9. The stripe voucher is a carrier class of its own and does not depend on any other carrier.
 - A card MAY carry a stripe voucher (this RFC) and a chip applet (RFC-003). The two MUST NOT share secrets.
 - The voucher MUST NOT be presented to or accepted by payment networks.
-

17. Testing Requirements

Implementations MUST include tests for:

- Track 1 and Track 3 round trip, wrong sentinel, wrong length, non-zero padding bits, wrong check characters;
 - wrong network code, version change;
 - double redemption (sequential) and concurrent double redemption (exactly one succeeds);
 - idempotent retry with the same nonce; replay with changed payload;
 - inactive, expired, revoked and locked vouchers;
 - PIN: correct, wrong, third failure locks, unlock;
 - register dump without k cannot redeem;
 - terminal counter regression, unregistered and revoked terminal;
 - payout retry after a service restart without a second payout;
 - face value above v_M rejected at issuance;
 - fuzzing of the track parser.
-

18. Reference Pseudocode

```
def redeem(req, svc):
    verify_terminal(req, svc)                # signature, registered, counter
    k = req.k
    C = H(enc(DOM_CL, k, svc.issuer_id))
    with svc.db.transaction():
        prev = svc.requests.get(req.request_id)
        if prev is not None:
            return prev.result if prev.payload == req.payload else E_REQUEST_REPLAY
        v = svc.register.get(C)
        if v is None:
            return E_VOUCHER_UNKNOWN
        if v.state == "REDEEMED":
            svc.flag_clone(C, req); return E_VOUCHER_CONSUMED
        if v.state != "ACTIVE":
            return state_error(v.state)
        if v.valid_until < svc.chain_height:
            return E_VOUCHER_EXPIRED
        if v.pin_tag is not None:
            check_pin(v, req)                # may lock the voucher
        v.state = "REDEEMED"
        svc.queue_payout(v.stamp_id, req.payout_address, v.face_value)
        svc.requests.put(req.request_id, req.payload, OK)
    return OK
```

19. Open Decisions

1. The format code for the track 1 record (an issuer code, not the bank card code) and its confirmation against the deployed track standard.
2. Whether track 1 or track 3 is the default carrier.
3. Partial redemption and change vouchers.
4. PIN entry device, key management and the PIN-block format for voucher PINs.

5. Default values for ϵ_M , velocity limits and voucher validity.
 6. Whether activation is mandatory.
 7. Terminal registration and revocation governance.
 8. Whether register commitments are anchored on-chain for audit.
-

20. Conclusion

Magnetic stripe vouchers extend LSC to the cheapest and most common reader hardware. They are safe only within the limits stated here: a bounded face value, a single online redemption, a register that stores commitments and detection of every second attempt. They complement NFC tokens (RFC-001) and chip cards (RFC-003) for cases where only a stripe reader is available.