

RFC-003: LanternSeaCoin Chip Card Extension

Status: Draft

Authors: LanternSeaCoin Protocol Team

Date: 6 October 2026

Language: English

Depends on: RFC-001 (NFC Extension), RFC-002 (Magnetic Stripe Voucher Extension, for the coexistence rules), LanternSeaCoin Reference Paper 2.0 (§14.1, §18, §19, §20)

Scope: LSC applet profile for chip cards (contact and contactless), command set, counter and credit model, PIN, card certificate, online reconciliation

1. Abstract

This RFC specifies an LSC applet profile for chip cards. The card is a secure element that holds a non-exportable Ed25519 key, a strictly monotonic counter, an offline credit counter with a limit, an optional PIN with a try counter, and a certificate from the card issuer.

The card does not hold coins. It authorizes redemption requests: it signs a request digest after incrementing its counter and consuming credit. This realizes the secure-element profile of RFC-001 on a standard chip card interface (ISO/IEC 7816 for contact cards, ISO/IEC 14443 for contactless).

An offline redemption authorized by a card is settled online as in RFC-001. A cloned card is detected when two different requests carry the same counter, and the loss it can cause is bounded by the credit limit.

2. Motivation

An NFC tag cannot hide data, keep a counter or limit its own use. A secure element can. A chip card makes the following possible where no phone is available:

- offline redemption up to a limit that the card itself enforces;
- a possession factor (the card) combined with a knowledge factor (the PIN) for larger amounts;
- detection of cloning through counter evidence instead of trust in the device;
- reuse of widely deployed card and reader infrastructure.

3. Terminology

Term	Meaning
Card	A chip card with an LSC applet
Applet	The LSC application on the card, selected by its AID
sk_d, pk_d	Card key pair; sk_d never leaves the card
card_id	$H(\text{enc}(\text{DOM_SE}, \text{pk_d}))$
Counter c_d	Strictly increasing 64-bit value, incremented for every signed request
Credit u_d	Offline amount consumed since the last reset
Limit U_max	Maximum offline credit; default 500 LSC
Attestation	Card signature over counter, amount and request digest
Card certificate	Issuer signature binding card_id, pk_d, U_max and validity

Term	Meaning
Reset authorization	Service signature that lowers u_d after online reconciliation
PED	PIN entry device (certified)
APDU	Application protocol data unit of ISO/IEC 7816-4
Terminal	Reader plus host software that talks to the card
H, enc	BLAKE2b-256 and the length-prefixed encoding of Reference Paper §14.1.2

4. Goals

This RFC aims to provide:

- a small, fixed command set that fits common secure elements;
- a counter that cannot repeat, even after a power failure;
- an offline credit limit enforced by the card;
- optional PIN protection above a value threshold;
- a card certificate that lets terminals and services verify that a card is genuine;
- clone detection and a bounded loss;
- compatibility with RFC-001 requests and the online reconciliation flow.

5. Non-goals

This RFC does not define:

- an EMV payment application or access to one;
- storage of coins on the card;
- a replacement for the NFC token (RFC-001) or the stripe voucher (RFC-002);
- a card personalization system or a card management platform;
- a new cryptographic primitive;
- a consumer wallet or user interface standard.

6. Card Model

6.1 Persistent state

Object	Size	Purpose
sk_d	32 bytes	private key, generated on the card, non-exportable
pk_d	32 bytes	public key
c_d	uint64	counter
u_d	uint64	consumed credit in base units (10 ⁸ per LSC)
U_max	uint64	credit limit
r_d	uint32	reset counter
p_in, tries	verifier, uint8	optional PIN and remaining attempts (default 3)
K_srv	32 bytes	service public key for reset and unblock authorizations
cert	about 145 bytes	card certificate

Object	Size	Purpose
flags	1 byte	PIN present, blocked, personalization locked

6.2 Identity

`card_id = H( enc( DOM_SE, pk_d ) )` `DOM_SE = "LSC-NFC-SE-v1"`

6.3 Card certificate

The card issuer signs the certificate with its issuer key `K_iss` (Reference Paper §14.1):

```
cert = Sign( K_iss, "LSC-CARD-CERT-v1" || enc( card_id, pk_d, U_max, valid_until,
    ↪ profile_version ) )
```

A terminal or service **MUST** verify `cert` against the issuer key before trusting an attestation. Cards can be revoked individually (§11).

6.4 Lifecycle

```
PERSONALIZED --activate--> ACTIVE --3 wrong PINs--> BLOCKED --unlock--> ACTIVE
ACTIVE | BLOCKED --revoke--> REVOKED
ACTIVE --valid_until passed--> EXPIRED
```

Personalization generates `sk_d` on the card, writes `cert`, `K_srv`, `U_max` and the PIN, and then locks personalization. A locked card **MUST NOT** accept personalization commands.

7. Application Selection and Interface

7.1 Application identifier

The applet is selected by a proprietary AID. This RFC proposes `F0 4C 53 43 01` (F0 proprietary prefix, ASCII LSC, version 01). The AID **MAY** be registered; until then it is proprietary.

7.2 Interfaces

The applet **MUST** work on the contact interface (ISO/IEC 7816, T=0 or T=1). It **MAY** be offered on the contactless interface (ISO/IEC 14443 type A or B) with identical behaviour. The contactless use is the secure-element mode of RFC-001.

7.3 Multi-application cards

The applet **MAY** coexist with an EMV payment applet on one chip. It **MUST** use its own AID and **MUST NOT** read or modify objects of any other applet. Payment data never enters LSC software (Reference Paper §19.1, R1). Loading the applet onto a multi-application card requires an arrangement with the card platform owner.

7.4 Reader requirements

The reader **MUST** allow proprietary APDUs (a transparent APDU interface, for example in PC/SC class readers). Readers that expose only an EMV payment kernel **MAY** refuse proprietary commands; such a reader is unsuitable. Reader suitability **MUST** be verified per product. PIN entry **MUST** use a certified PED.

8. Command Set

8.1 Commands

Command	CLA INS	P1 P2	Command data	Response data
SELECT	00 A4	04 00	AID	profile version, card_id
GET DATA	00 CA	tag	none	data object (\$9)
VERIFY	00 20	00 01	PIN block	none
GET STATE	80 10	00 00	none	pk_d(32), c_d(8), u_d(8), U_max(8), flags(1)
SIGN REQUEST	80 40	00 00	D(32), amount(8)	c'(8), signature(64)
RESET CREDIT	80 50	00 00	r'(4), u_new(8), service signature(64)	none
UNBLOCK PIN	80 52	00 00	r'(4), new PIN block, service signature(64)	none

All integers are little-endian.

8.2 SIGN REQUEST

Input is the request digest D and the amount. The card MUST:

1. check that personalization is locked and the state is ACTIVE;
2. check amount against the per-token maximum (100 LSC);
3. check $u_d + \text{amount} \leq U_{\text{max}}$;
4. if $\text{amount} \geq v_{\text{PIN}}$ (reference value 20 LSC), require a successful VERIFY in the current session;
5. compute $c' = c_d + 1$ and $u' = u_d + \text{amount}$;
6. **commit c' and u' to non-volatile memory in one atomic transaction;**
7. only then compute and return the attestation.

$\text{attestation} = \text{Sign}(sk_d, \text{"LSC-NFC-SE-v1"} || \text{enc}(card_id, c', \text{amount}, D))$

If power fails after step 6 and before the response, the next call yields a larger counter. Gaps are allowed. The consumed credit of an unfinished call is restored by the next reset.

8.3 RESET CREDIT

The card verifies the service signature with K_{srv} over $\text{enc}(\text{"LSC-NFC-RESET-v1"}, card_id, r', u_new)$. It MUST require $r' = r_d + 1$ and $u_new \leq u_d$, then set $u_d = u_new$ and $r_d = r'$. The reset counter prevents replay of old authorizations.

8.4 UNBLOCK PIN

The service signs $\text{enc}(\text{"LSC-NFC-UNBLOCK-v1"}, card_id, r', pin_verifier)$. The card sets the new PIN, restores the try counter and increments r_d . Only the service can unblock a card.

8.5 Status words

SW	Meaning	Terminal error
9000	success	none
63Cx	PIN wrong, x attempts left	E_PIN_WRONG
6983	PIN blocked	E_PIN_BLOCKED
6982	security status not satisfied (PIN or signature)	E_PIN_REQUIRED or E_SIGNATURE
6985	conditions of use not satisfied (credit, limit, state)	E_CREDIT
6A80	incorrect data	E_APDU
6A82	applet not found	E_CARD_UNKNOWN
6700	wrong length	E_APDU
6D00, 6E00	INS or CLA not supported	E_APDU

9. Data Objects

Tag	Object	Size
DF01	cert	about 145 bytes
DF02	profile version	1 byte
DF03	card_id	32 bytes
DF04	valid_until	8 bytes
DF05	remaining PIN attempts	1 byte

sk_d has no tag and no command returns it.

10. Redemption

10.1 Offline redemption with a card

1. The terminal reads the NFC token or receives it by QR (RFC-001 §6).
2. The terminal computes the request digest $D = H(\text{enc}(\text{DOM_REQ}, \text{net}, \text{token_id}, \text{stamp_id}, \text{redeemer}, \text{amount}))$.
3. If required, the user enters the PIN on the PED; the terminal sends VERIFY.
4. The terminal sends SIGN REQUEST with D and amount.
5. The terminal builds the RFC-001 redeem request. As nonce it uses the attestation reference se:<card_id>:<c'> (the convention of this RFC for card-attested requests); the request_id is constructed as recommended in RFC-001 §7.2.
6. The terminal stores the request before it reports success; if storing fails it MUST report failure, not success.

10.2 Online reconciliation

The service MUST verify:

1. cert against the issuer key and the revocation list;
2. the attestation signature under pk_d;
3. the counter: strictly greater than the highest counter seen for card_id;
4. the same (card_id, c') with a different D is a clone indicator → E_REQUEST_REPLAY and card review;
5. credit accounting: the sum of unsettled amounts per card MUST NOT exceed U_max;
6. the redemption itself as in RFC-001 §8.

10.3 Credit reset

After successful reconciliation the service issues a reset authorization (§8.3) for the settled amount. The card stays usable offline only up to $U_{\max}$ between resets.

10.4 Bound on loss

If an attacker creates c clones with identical state, each clone can spend the remaining credit before reconciliation:

$$L_{\text{clone}} \leq c * (U_{\max} - u_0)$$

A clone is detected as soon as two different requests with the same counter reach the service. Above v_{PIN} the PIN limits use by a thief who holds a genuine card.

11. Card Revocation

The service keeps a revocation list of `card_id` values. A card is revoked on loss report, clone evidence or issuer decision. Terminals SHOULD refresh the list whenever they are online; the service MUST check it at reconciliation. Entries are only added; the merge is monotonic and idempotent, and removal needs a governance decision.

12. Consensus and Governance Rules

Online settlement MUST obey the consensus rules. Cards, certificates and revocation lists are issuer and service systems and are not part of consensus. This RFC does not authorize anyone to alter canonical state. Corrections follow the existing governance mechanism (Reference Paper §9).

13. Error Codes

In addition to the codes of RFC-001 §13:

Code	Meaning
<code>E_CARD_UNKNOWN</code>	Applet not found or card not recognized
<code>E_CARD_REVOKED</code>	Card on the revocation list
<code>E_CARD_CERT</code>	Certificate invalid or expired
<code>E_COUNTER</code>	Counter not strictly increasing
<code>E_CREDIT</code>	Offline credit exhausted or limit exceeded
<code>E_PIN_REQUIRED</code>	PIN needed for this amount
<code>E_PIN_WRONG</code>	PIN incorrect; remaining attempts are returned
<code>E_PIN_BLOCKED</code>	PIN blocked
<code>E_APDU</code>	Card returned an unexpected status word
<code>E_READER</code>	Reader does not support proprietary APDUs

14. Security Considerations

14.1 Cloning

sk_d is generated on the card and cannot be read. A clone therefore needs a hardware attack. A copied state without the key cannot sign. The counter makes any successful clone detectable at reconciliation, and L_clone bounds the loss.

14.2 Tearing and counter rollback

The counter and credit MUST be committed atomically before the attestation leaves the card. A card platform that cannot guarantee atomic non-volatile updates MUST NOT be used. Fault injection that rolls back the counter is covered by the loss bound and by service-side counter checks.

14.3 Relay attacks

On the contactless interface a relay can forward APDUs from a genuine card to a remote terminal. The counter, the credit limit and, above v_PIN, the PIN bound the damage. This RFC defines no distance bounding.

14.4 PIN

The PIN is checked inside the card with a try counter. It MUST be entered on a PED. Terminal software MUST NOT log PIN data.

14.5 Terminal compromise

A compromised terminal can request signatures within the card's credit and cannot extract the key. Without the PIN it cannot sign amounts above v_PIN.

14.6 Side channels and platform quality

Implementations SHOULD use secure elements that are independently evaluated (for example under Common Criteria) and resistant to side-channel and fault attacks.

14.7 Supply chain and personalization

Key generation and personalization MUST occur in a controlled environment. Personalization commands MUST be rejected after the lock.

15. Privacy

card_id is a stable identifier. Anyone who can read the card can link its uses. Issuers SHOULD disclose this. The card holds no name or account data. Logs at terminal and service fall under data-protection law and MUST have retention limits.

16. Compatibility

- The request format, nonce convention and online flow are those of RFC-001 §7 and §8; the card is the secure element of RFC-001's SE profile.
- The value rule and the loss bound for cloned devices are those of Reference Paper §20.6 and §20.9; the amount limits of this RFC (per-token maximum, U_max, v_PIN) are defined in §6 and §8.

- The applet MAY coexist with an RFC-002 stripe on the same card but MUST NOT share secrets with it.
- Profile versioning: profile_version in the certificate and DF02 selects the rules; unknown versions MUST be rejected.

17. Testing Requirements

Implementations MUST include tests for:

- SELECT with a correct and a wrong AID;
- SIGN REQUEST: counter increases by one per call; u_d accumulates; limit exceeded gives 6985; amount above the token maximum rejected;
- atomicity: power cut before and after the commit never repeats a counter value (tearing test, at least 10^6 cycles);
- PIN: correct, wrong, third failure blocks, unblock with a valid and an invalid service signature;
- reset replay: an old r' is rejected;
- certificate: valid, expired, revoked, wrong issuer, tampered fields;
- attestation verification at the service for valid and altered data;
- clone simulation: two requests with the same counter and different digests are flagged;
- coexistence with another applet: no access across applets;
- fuzzing of the APDU parser (lengths, unknown INS, truncated data);
- interoperability: a complete offline redemption reconciled by an RFC-001 service.

18. Reference Pseudocode

```
def sign_request(card, D, amount, session):
    assert card.state == "ACTIVE" and card.locked
    if amount > PER_TOKEN_MAX: return SW_6985
    if card.u + amount > card.U_max: return SW_6985
    if amount >= V_PIN and not session.pin_ok: return SW_6982
    c_new, u_new = card.c + 1, card.u + amount
    card.nvm_commit(c=c_new, u=u_new) # atomic, before any output
    msg = enc("LSC-NFC-SE-v1", card.card_id, c_new, amount, D)
    return (c_new, ed25519_sign(card.sk, msg))

def verify_attestation(svc, card_cert, c_new, amount, D, sig):
    svc.check_cert(card_cert) # issuer signature, validity,
    ↪ revocation
    ok = ed25519_verify(card_cert.pk, enc("LSC-NFC-SE-v1", card_cert.card_id,
    ↪ c_new, amount, D), sig)
    if not ok: return E_SIGNATURE
    seen = svc.counters.get(card_cert.card_id)
    if seen is not None and c_new <= seen.max_counter:
        if seen.digest_at(c_new) not in (None, D):
            ↪ svc.flag_clone(card_cert.card_id); return E_REQUEST_REPLAY
        if seen.digest_at(c_new) == D: return svc.stored_result(...)
    svc.counters.record(card_cert.card_id, c_new, D, amount)
    return OK
```

19. Open Decisions

1. Ed25519 support on the chosen card platform. If it is missing, a second profile with ECDSA P-256 MAY be defined; it is not part of this RFC.
 2. Registration of the AID.
 3. Default values for U_max, the per-token maximum and v_PIN.
 4. Certificate chain depth (single issuer key or an intermediate level).
 5. Whether the card keeps a token store, which would need an on-card signature check.
 6. Revocation list distribution to offline terminals.
 7. Policy for lost-credit cases after interrupted calls.
 8. Multi-application deployment rules with payment applets.
 9. Distance-bounding or other relay protection for the contactless interface.
-

20. Conclusion

The chip card turns the secure-element profile of RFC-001 into a deployable carrier: a non-exportable key, an atomic counter, a credit limit enforced on the card and an optional PIN. Clones are detected by counter evidence and their damage is bounded. Together with RFC-001 (NFC tokens) and RFC-002 (magnetic stripe vouchers) it covers the three physical carriers of the card and terminal layer in Reference Paper §19.