Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 17 July 2026

A. Boell, Ed.
Midwest Cyber LLC

13 January 2026

Digital Signing of Physical Items Protocol (DSPIP)
draft-midwestcyber-dspip-01

Abstract

This document specifies the Digital Signing of Physical Items Protocol (DSPIP), a cryptographic protocol for authenticating physical items using digitally signed QR codes. This specification focuses on the SHIP type for shipping and logistics applications, providing cryptographic authentication of packages with chain of custody tracking.

The protocol introduces privacy-preserving delivery mechanisms including encrypted recipient information, last mile provider selection, physical anti-cloning protections through split-key labels, and scalable revocation and delivery confirmation systems. DSPIP uses DNS-based public key distribution similar to DKIM and supports optional blockchain integration for immutable record keeping. While this specification focuses on shipping applications, the protocol includes a type field to enable future expansion to other use cases.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 17 July 2026.

Table of Contents

1. Introduction

Supply chain logistics require verifiable proof of package authenticity, chain of custody, and delivery confirmation. Current shipping systems rely on proprietary tracking databases with varying security models and no cryptographic verification of package authenticity or delivery. QR codes on shipping labels can be easily copied, leading to fraud and misdirection of packages.

DSPIP provides a cryptographic protocol specifically designed for shipping and logistics, enabling verification of package origin, custody chain, and delivery while protecting recipient privacy. The protocol addresses the fundamental challenge of QR code cloning through physical binding mechanisms and provides privacy-preserving delivery that reveals recipient information only to authorized last mile providers.

By using the secp256k1 elliptic curve, DSPIP keys are compatible with major blockchain networks, allowing optional integration for immutable custody records while not requiring blockchain for basic operation. The protocol includes a type field set to "SHIP" for this specification, enabling future expansion to other applications.

The design goals for DSPIP shipping protocol are:

* Cryptographic Authentication: Verify package origin and integrity through digital signatures

* Privacy Preservation: Protect recipient information during transit through encryption

* Anti-Cloning: Prevent QR code duplication through physical binding mechanisms

* Decentralized Verification: No central authority required for package validation

* Offline Capable: Verification possible without network connectivity

* Delivery Confirmation: Cryptographic proof of successful delivery

* Scalable Operations: Support high-volume shipping with efficient revocation and confirmation systems

* Standards Compliant: Integrate with existing logistics systems and standards

* Cost Effective: Near-zero marginal cost per package authenticated

* Blockchain Optional: Can integrate with blockchain but operates independently

DSPIP follows a "digital envelope" paradigm for shipping labels:

* Envelope exterior (publicly readable): Sender identity, last mile provider destination, tracking number, and timestamp are encoded but not encrypted. This functions like the address on a physical envelope.

* Signature (authenticity seal): A cryptographic signature proves the label was created by the claimed sender and has not been tampered with, similar to a shipping seal.

* Private contents (encrypted): The actual recipient address and delivery instructions are encrypted for the last mile provider, like the contents of a sealed package that only the authorized recipient can access.

This model protects recipient privacy while maintaining package routability and authenticity verification.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Relationship to Existing Standards

DSPIP builds upon established standards while introducing novel shipping-specific features:

Standards This Document Builds Upon:

* [RFC6376] (DKIM): DSPIP adopts DKIM’s DNS-based public key distribution model for shipping providers

* [RFC8017] (PKCS #1): Uses established cryptographic specifications

* [ISO18004]: Utilizes QR code standards for label encoding

* [SEC2]: Employs secp256k1 curve specifications (same curve as used by Bitcoin [Bitcoin] and Ethereum [Ethereum])

* [ECIES]: Elliptic Curve Integrated Encryption Scheme for recipient privacy

* [Ed25519]: For split-key label authentication

Related Shipping Standards:

* [GS1-EPCIS]: DSPIP adds cryptographic authentication to event-based visibility

* [EDI856] (ASN): DSPIP enables verification without central database

* Last Mile Standards: DSPIP adds encrypted routing for privacy

Novel Contributions:

* Privacy-Preserving Delivery: Encrypted recipient information with last mile provider routing

* Physical Anti-Cloning: Split-key labels with destructive reveal prevent QR code duplication

* Cryptographic Proof of Delivery: Challenge-response confirmation without key exposure

* Scalable Confirmation Systems: Bulk revocation and delivery lists for high-volume operations

4. Protocol Overview

4.1. Data Flow

The DSPIP shipping protocol flow consists of the following steps:

1. Payload Creation: Sender creates a JSON payload containing sender information, last mile provider or recipient, tracking number, and timestamp

2. Privacy Selection: Choose privacy mode (standard, encrypted, or split-key) based on security requirements

3. Recipient Encryption: For privacy modes, encrypt recipient information with last mile provider’s public key

4. Encoding: Payload is Base64 encoded to ensure consistent handling

5. Signing: Sender signs the data using their private key (or label’s private key for split-key mode)

6. QR Generation: Complete data structure is serialized and encoded as a QR code

7. Label Application: QR code is printed on shipping label and attached to package

8. Transit Scanning: Carriers scan QR code at each custody transfer point

9. DNS Lookup: Scanner queries DNS TXT record to retrieve public key

10. Verification: Scanner verifies signature (or requests Zone B reveal for split-key)

11. Privacy Decryption: Last mile provider decrypts recipient information using their private key

12. Delivery: Package delivered to actual recipient address

13. Confirmation: Cryptographic proof of delivery recorded

14. Optional Blockchain: Custody chain recorded for audit trail

4.2. Protocol Workflow

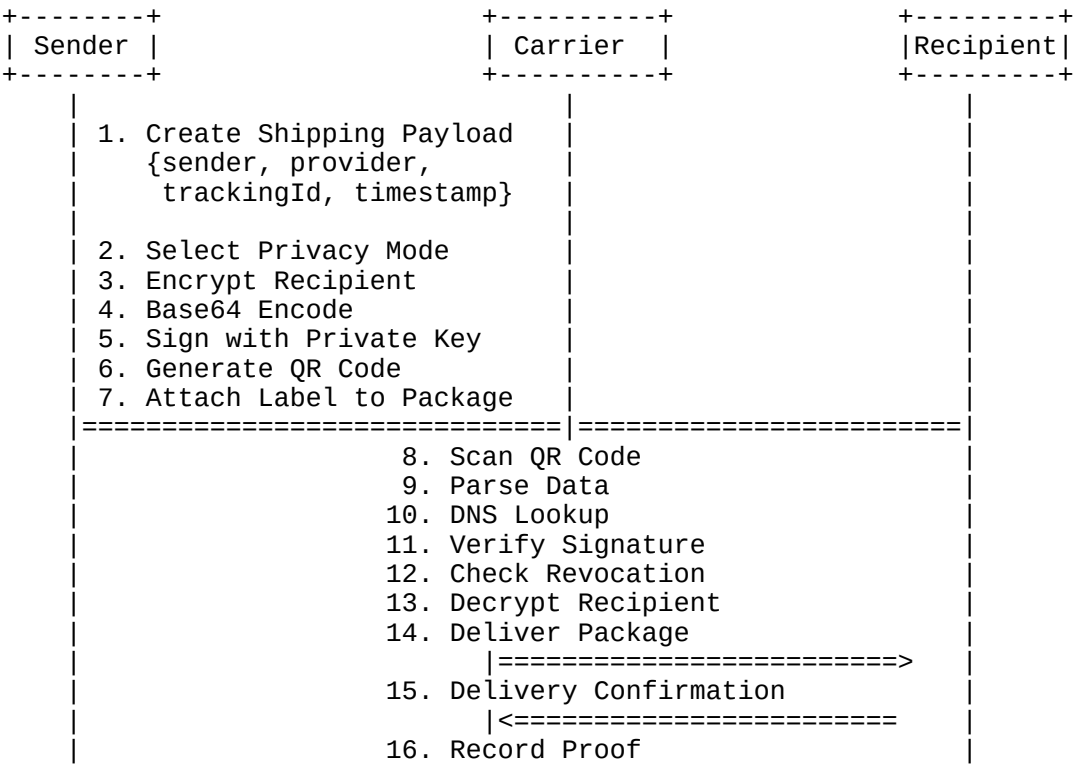


Figure 1: DSPIP Shipping Flow

5. Data Formats

5.1. Payload Structure

The payload contains shipping information encoded as JSON. Country codes use [ISO3166] alpha-2 format:

```
{
  "type": "SHIP",               // REQUIRED: Fixed for shipping
  "issuer": {                   // REQUIRED: Sender information
    "name": "string",
    "organization": "string",
    "address": {
      "street1": "string",
      "city": "string",
      "state": "string",
      "postalCode": "string",
      "country": "string"       // REQUIRED: per ISO3166 alpha-2
    }
  },
  "subject": {                  // REQUIRED: Recipient/Provider
    "name": "string",
    "lastMileProvider": "string",
    "address": { ... }
  },
  "itemId": "string",           // REQUIRED: Tracking number
  "timestamp": number,          // REQUIRED: Unix timestamp (ms)
  "typeData": {}                // SHIP-specific fields
}
```

5.2. SHIP Type Payload Fields

The typeData object contains shipping-specific information including privacy mode, carrier details, customs information, and delivery confirmation settings.

5.3. Privacy Modes

DSPIP supports three privacy modes for shipping:

5.3.1. Standard Mode (Legacy)

Traditional shipping with full recipient information visible.

5.3.2. Encrypted Mode

Privacy-preserving with encrypted recipient data. The recipient’s address and delivery instructions are encrypted with the last mile provider’s public key.

5.3.3. Split-Key Mode

Maximum security with physical anti-cloning. Uses Ed25519 keys printed in separate scratch-off zones on the physical label.

5.4. QR Data Structure

The complete QR code data structure contains:

* protocol: Fixed string "DSPIP"
* version: Semantic version string ("1.0")
* type: Fixed to "SHIP" for this specification
* keyLocator: DNS path for public key lookup
* encodedPayload: Base64-encoded JSON payload
* signature: Hex-encoded ECDSA signature
* privateMessage: Optional Base64-encoded encrypted message

5.5. Serialization Format

QR data MUST be serialized using pipe (|) delimiters:

DSPIP|<version>|<type>|<keyLocator>|<payload>|<sig>[|<msg>]

The pipe delimiter was chosen for its low frequency in Base64 and domain names. Implementations MUST validate that exactly 6 or 7 pipe-delimited fields are present.

6. DNS Key Distribution

DSPIP uses DNS TXT records for public key distribution, following the model established by DKIM [RFC6376].

6.1. Key Locator Format

Key locators MUST follow this pattern:

<selector>._dspip.<domain>

Where selector is a unique identifier for the specific key, _dspip is the fixed protocol subdomain (underscore prefix per [RFC8552]), and domain is the organization’s domain name.

6.2. DNS TXT Record Format

The DNS TXT record MUST contain a semicolon-separated list of tag=value pairs:

v=DSPIP1; k=ec; c=secp256k1; p=<base64_public_key>; [optional]

Required tags: v (protocol version), k (key type), c (curve identifier), p (public key in Base64).

Optional tags: t (creation timestamp), x (expiration), n (notes), types, auth, address, coverage.

6.2.1. Address Field Format

The address field specifies the physical location of signing facilities, primarily for last mile providers. This enables verifiers to display facility location on shipping labels for encrypted privacy mode, where the recipient address is hidden.

The address field uses a scheme prefix to indicate the encoding:

* plus: (DEFAULT) Open Location Code / Plus Code
* street: Percent-encoded street address per [RFC3986]
* geo: Geographic coordinates (latitude,longitude)
* facility: Named facility reference

If no scheme prefix is present, implementations MUST interpret the value as a Plus Code (plus: scheme).

6.2.1.1. Plus Code Scheme (Default)

Plus Codes (Open Location Codes) are the RECOMMENDED addressing scheme due to their compact representation, open standard licensing, and global coverage.

Format: address=plus:<plus_code>

Short form: address=<plus_code>

Examples:

```
address=plus:86HJW222+22        (full Plus Code, ~3m precision)
address=849VCWC8+R9             (implicit plus: scheme)
address=plus:CWC8+R9,Omaha      (short code with locality)
```

Plus Codes are 10-11 characters for full precision or 6-7 characters when combined with a locality reference. See [PLUSCODE] for the Open Location Code specification.

6.2.1.2. Street Address Scheme

Traditional street addresses encoded using percent-encoding per [RFC3986].

Format: address=street:<percent_encoded_address>

Examples:

address=street:1234%20Post%20Office%20Way%2C%20Omaha%2C%20NE%2068101
address=street:123%20Business%20Ave%2C%20Suite%20100

Street addresses SHOULD include city and postal code for clarity.

6.2.1.3. Geographic Coordinate Scheme

Decimal latitude and longitude coordinates.

Format: address=geo:<latitude>,<longitude>

Examples:

address=geo:41.2565,-95.9345
address=geo:40.7128,-74.0060

Coordinates MUST use decimal degrees with negative values for South and West. Precision SHOULD be at least 4 decimal places (~11 meter accuracy).

6.2.1.4. Facility Reference Scheme

Named facility reference for organizations with published facility directories.

Format: address=facility:<facility_id>

Examples:

address=facility:USPS-OMAHA-MAIN
address=facility:UPS-NE-401

Facility references require external directory lookup. Organizations using this scheme SHOULD publish facility directories at a well-known location or via the discovery record.

6.2.1.5. Address Field Requirements

* The address field is OPTIONAL for key records
* Last mile providers SHOULD include address for encrypted mode label display
* Implementations SHOULD prefer Plus Codes for new deployments
* Verifiers MUST support all four schemes
* When displaying addresses, verifiers MAY render Plus Codes as clickable links to mapping services

6.3. Selector Discovery

To enable discovery of available selectors for a domain, organizations MAY publish a discovery record at the base _dspip subdomain.

6.3.1. Discovery Record Format

```
_dspip.example.com. IN TXT
  "v=DSPIP1; type=discovery;
   selectors=warehouse,fulfillment,returns;
   delegation=list"
```

Discovery record tags:

* v: Protocol version (MUST be "DSPIP1")
* type: Record type (MUST be "discovery" for discovery records)
* selectors: Comma-separated list of available selectors
* delegation: Delegation scheme for selector organization

6.3.2. Delegation Schemes

The delegation tag indicates how selectors are organized:

* list: Explicit enumeration in selectors field
* geo: Geographic prefixes (e.g., us-west, eu-central)
* postal: Postal code prefixes (e.g., 68_, 90_)
* region: Regional identifiers (e.g., midwest, northeast)

* service: Service-based selectors (e.g., express, ground)

Examples:

List delegation:
_dspip.example.com. IN TXT "v=DSPIP1; type=discovery;
  selectors=warehouse-a,warehouse-b,fulfillment; delegation=list"

Geographic delegation:
_dspip.globalcorp.example. IN TXT "v=DSPIP1; type=discovery;
  delegation=geo; pattern=<region>-<facility>"

Postal code delegation:
_dspip.usps.example. IN TXT "v=DSPIP1; type=discovery;
  delegation=postal; pattern=<zip5>; coverage=00000-99999"

6.3.3. Discovery Requirements

* Discovery records are OPTIONAL

* Verifiers SHOULD NOT require discovery records for verification

* Discovery records SHOULD be cached according to DNS TTL

* Implementations MAY use discovery for user interfaces and
  directory services

6.4. Key Lifecycle Management

DSPIP keys have a defined lifecycle to ensure security while
maintaining verification capability for in-transit packages.

6.4.1. Key Record Fields

The DNS TXT record supports the following lifecycle fields:

Required fields:

* v: Protocol version (MUST be "DSPIP1")

* k: Key type (MUST be "ec")

* c: Curve identifier (MUST be "secp256k1")

* p: Public key in Base64 encoding

SHOULD include:

* t: Key creation timestamp (Unix time). If omitted, verifiers SHOULD treat the key as valid from an indefinite past.

* exp: Signing expiration timestamp (Unix time). After this time, the key MUST NOT be used to sign new items. If omitted, the key has no signing expiration.

MAY include:

* exp-v: Verification expiration timestamp (Unix time). After this time, the key MUST NOT be used to verify any signatures. If omitted, defaults to exp value (if present) or no expiration.

* s: Key status. Values: "active" (default if omitted), "verify-only" (can verify but not sign), "revoked" (completely invalid).

* seq: Sequence number for key updates. Higher values supersede lower values for the same selector.

* n: Human-readable notes (percent-encoded)

6.4.2. Key Lifecycle States

Keys transition through the following states:

1. Active: Key can sign new items and verify existing signatures. Status: s=active (or omitted), current time < exp

2. Verify-Only: Key cannot sign new items but can verify existing signatures. Status: s=verify-only, or exp < current time < exp-v

3. Expired: Key is completely invalid and MUST NOT be used. Status: s=revoked, or current time > exp-v

6.4.3. Recommended Key Lifetime

For shipping applications, the following key lifetime is RECOMMENDED:

* Active signing period: 365 days (1 year)

* Verify-only period: 365 additional days (1 year)

* Total key lifetime: 730 days (2 years)

This provides sufficient time for packages to complete transit (typically 1-30 days) while limiting exposure from compromised keys.

Example DNS record with lifecycle fields:

```
warehouse._dspip.example.com. IN TXT "v=DSPIP1; k=ec; c=secp256k1;
  p=AzmjYBMwFZfa70H75ZOgLMUT0LVVJ+wt8QUOLo/0nIXC;
  t=1703548800; exp=1735084800; exp-v=1766620800;
  s=active; seq=1; n=Main%20Warehouse%20Key%202025"
```

6.4.4. Key Rotation

Organizations SHOULD implement key rotation:

* Generate new key before current key’s exp timestamp
* Publish new key with incremented seq value
* Transition signing to new key
* Maintain old key for verification until exp-v
* Remove old key after exp-v passes

6.4.5. Field Omission Semantics

When lifecycle fields are omitted, verifiers MUST apply these rules:

* t omitted: Key has no creation date constraint
* exp omitted: Key has no signing expiration (signs indefinitely)
* exp-v omitted: Defaults to exp value; if exp also omitted, key verifies indefinitely
* s omitted: Key is "active"
* seq omitted: Treated as seq=0; cannot supersede keys with seq > 0

Note: While omitting fields is permitted for backwards compatibility, implementations SHOULD include t, exp, and exp-v for production keys to enable proper lifecycle management.

6.4.6. Record Signature (rsig)

The rsig field provides cryptographic authentication of lifecycle metadata within the DNS record itself. This protects against unauthorized modification of lifecycle fields (t, exp, exp-v, s, seq) by intermediate DNS resolvers or caches.

6.4.6.1. Threat Model

Without rsig, an attacker with access to an intermediate DNS resolver or cache could modify lifecycle fields without invalidating QR code signatures. For example:

* Extending exp to hide that a key has expired
* Changing s=revoked to s=active to hide key compromise
* Modifying seq to prevent key rotation from taking effect

The rsig field allows verifiers to detect such modifications by checking a signature over the lifecycle metadata.

6.4.6.2. Signable Content

The rsig signature covers a canonical string of lifecycle fields:

rsig_content = selector "|" t "|" exp "|" exp-v "|" s "|" seq

Where:

* selector: The selector portion of the key locator (e.g., "warehouse")
* t: Creation timestamp as decimal string, or empty if omitted
* exp: Signing expiration as decimal string, or empty if omitted
* exp-v: Verification expiration as decimal string, or empty if omitted
* s: Status value, or empty if omitted (defaults to "active")
* seq: Sequence number as decimal string, or empty if omitted

Example signable content:
"warehouse|1703548800|1735084800|1766620800|active|1"

6.4.6.3. Signature Generation

The rsig value is generated as follows:

1. Construct the signable content string as defined above
2. Compute SHA-256 hash of the UTF-8 encoded signable content

3. Sign the hash using ECDSA with the same private key that corresponds to the public key in the p= field

4. Encode the signature in Base64

The same key signs both QR codes and the rsig, providing proof that the key owner authorized the lifecycle metadata.

6.4.6.4. Verification Process

When rsig is present, verifiers SHOULD:

1. Extract the rsig value from the DNS record

2. Reconstruct the signable content from the record’s lifecycle fields

3. Compute SHA-256 hash of the signable content

4. Verify the rsig signature using the public key from the p= field

5. If verification fails, treat the record as untrusted

6.4.6.5. Requirements

* rsig is OPTIONAL for DNS records

* When rsig is present, verifiers SHOULD verify it

* If rsig verification fails, verifiers SHOULD reject the record or flag it with a LIFECYCLE_UNVERIFIED warning

* Implementations MAY operate in "rsig-required" mode for high-security deployments

* rsig provides defense-in-depth; DNSSEC provides stronger guarantees when available

6.4.6.6. Relationship to DNSSEC

DNSSEC and rsig serve complementary purposes:

* DNSSEC: Zone owner signs all records; proves record came from authoritative zone

* rsig: Key owner signs lifecycle metadata; proves the signing key holder authorized the lifecycle values

For maximum security, deployments SHOULD use both DNSSEC and rsig. DNSSEC alone is sufficient if the organization controls their DNS infrastructure end-to-end.

6.5. Key Revocation

When a signing key is compromised or must be retired, organizations MUST have a mechanism to inform verifiers that the key should no longer be trusted. While the s=revoked status in the key record provides this capability, DNS caching can delay propagation.

The key revocation record provides a dedicated, frequently-checked endpoint for key status that operates independently of key record caching.

6.5.1. Key Revocation Record Format

```
_revoked-key._dspip.example.com. IN TXT
  "v=DSPIP1; type=key-revocation;
   selector=warehouse;
   revoked=1703548900;
   reason=compromised;
   replacement=warehouse-v2"
```

Fields:

* v: Protocol version (MUST be "DSPIP1")

* type: Record type (MUST be "key-revocation")

* selector: The selector of the revoked key (REQUIRED)

* revoked: Unix timestamp when the key was revoked (REQUIRED)

* reason: Reason for revocation (REQUIRED). Values: compromised (private key was exposed or stolen), retired (key reached end of lifecycle), superseded (key replaced by newer key), suspended (temporarily disabled, may be reactivated)

* replacement: Selector of the replacement key (OPTIONAL)

6.5.2. Multiple Key Revocations

Organizations with multiple revoked keys MAY use multiple TXT records or a bulk format:

```
 Individual records:
 _revoked-key._dspip.example.com. IN TXT "v=DSPIP1; type=key-revocation;
   selector=warehouse-a; revoked=1703548900; reason=compromised"
 _revoked-key._dspip.example.com. IN TXT "v=DSPIP1; type=key-revocation;
   selector=warehouse-b; revoked=1703549000; reason=retired"

 Bulk pointer:
 _revoked-key._dspip.example.com. IN TXT
   "v=DSPIP1; type=key-revocation-list;
    url=https://example.com/dspip/revoked-keys.json;
    updated=1703548800"
```

6.5.3. Verifier Requirements for Key Revocation

Verifiers MUST check key revocation status before trusting signatures:

1. Query _revoked-key._dspip.<domain> with minimal or no caching

2. If the key's selector appears in the revocation record(s), REJECT the signature immediately

3. Key revocation takes precedence over s=active in the key record

4. Verifiers SHOULD use short TTLs (60-300 seconds) when caching key revocation records

5. For high-security deployments, verifiers MAY query key revocation on every verification without caching

6.5.4. Relationship to Key Record Status

The s (status) field in key records and _revoked-key records serve complementary purposes:

* s=revoked in key record: Authoritative status, but subject to DNS caching delays

* _revoked-key record: Dedicated revocation channel with shorter cache lifetime for faster propagation

Organizations SHOULD update both when revoking a key:

1. Publish _revoked-key record (immediate propagation)

2. Update key record with s=revoked (authoritative status)

Verifiers MUST treat a key as revoked if EITHER indicates revocation.

6.6. Package Revocation

Individual packages may need to be revoked due to loss, theft, or fraud without revoking the signing key. DSPIP provides two mechanisms for package revocation.

6.6.1. Individual Package Revocation Record

For low-volume revocations, organizations MAY publish individual revocation records directly in DNS:

```
_revoked._dspip.example.com. IN TXT
  "v=DSPIP1; type=item-revocation;
   itemId=TRACK-2025-000123;
   revoked=1703548900;
   reason=lost"
```

Fields:

* v: Protocol version (MUST be "DSPIP1")

* type: Record type (MUST be "item-revocation")

* itemId: The revoked package identifier (REQUIRED)

* revoked: Unix timestamp when the package was revoked (REQUIRED)

* reason: Reason for revocation (REQUIRED). Values: lost (package lost in transit), stolen (package was stolen), damaged (package damaged and reshipped), recalled (package recalled by sender), fraud (fraudulent QR code detected)

* replacementId: New itemId for reshipped package (OPTIONAL)

6.6.2. Bulk Package Revocation

For scalability, senders MAY publish bulk revocation lists:

6.6.2.1. Revocation Pointer Record

```
revocation._dspip.example.com. IN TXT
  "v=DSPIP1; type=revocation;
   url=https://example.com/dspip/revoked.json;
   format=json|bloom;
   updated=1703548800;
   ttl=86400"
```

6.6.2.2. Revocation List Format

```
{
  "version": "1.0",
  "issuer": "warehouse._dspip.example.com",
  "updated": 1703548800,
  "expires": 1719316800,
  "revoked": [
    {
      "itemId": "TRACK-2025-000123",
      "revoked": 1703548900,
      "reason": "lost|stolen|damaged|recalled",
      "expires": 1719316900,
      "replacementId": "TRACK-2025-000124"
    }
  ],
  "signature": "issuer_signature"
}
```

Bloom filter format (for privacy):

```
{
  "version": "1.0",
  "issuer": "warehouse._dspip.example.com",
  "updated": 1703548800,
  "filter": {
    "type": "bloom",
    "size": 1048576,
    "hashes": 7,
    "falsePositiveRate": 0.001,
    "data": "base64_bloom_filter_data"
  },
  "signature": "issuer_signature"
}
```

6.6.2.3. Revocation Requirements

* Lists MUST be served over HTTPS
* Lists SHOULD auto-prune entries older than 180 days
* Lists MAY use [BLOOM] filters for privacy
* Verifiers SHOULD cache lists based on TTL

6.6.2.4. Revocation Freshness

To ensure timely revocation detection, revocation records MUST NOT be cached indefinitely. The revocation list format includes freshness fields:

```
{
  "version": "1.0",
  "issuer": "warehouse._dspip.example.com",
  "ts": 1703548800,
  "max-age": 900,
  "updated": 1703548800,
  "revoked": [...]
}
```

Freshness field semantics:

* ts: Unix timestamp when the revocation list was generated. MUST be present. Verifiers MUST reject lists where current_time - ts exceeds max-age.

* max-age: Maximum age in seconds before the list must be refreshed. MUST be present. Recommended values: 300-900 seconds (5-15 min) for active verification, up to 3600 seconds (1 hour) for background checks.

Verifier requirements:

* Verifiers MUST check ts + max-age > current_time before using a cached revocation list

* Verifiers MUST refresh revocation lists when max-age expires

* Verifiers SHOULD NOT serve stale revocation data to end users

* Verifiers MAY implement background refresh to maintain fresh data

Note: These freshness requirements apply to revocation lists only. Key records follow standard DNS TTL caching semantics.

6.7. Delivery Confirmation Distribution

Last mile providers MAY publish delivery confirmations:

6.7.1. Delivery Confirmation Pointer

```
delivered._dspip.usps.example. IN TXT
  "v=DSPIP1; type=delivered;
   url=https://usps.example/dspip/delivered.json;
   updated=1703548800;
   ttl=3600"
```

6.7.2. Delivery Confirmation Format

```
{
  "version": "1.0",
  "provider": "omaha-main._dspip.usps.example",
  "updated": 1703548800,
  "expires": 1711324800,
  "delivered": [
    {
      "itemId": "TRACK-2025-000123",
      "deliveredAt": 1703548900,
      "deliveredTo": {
        "type": "signature|photo|cryptographic",
        "data": "base64_proof_data",
        "recipientKeyHash": "sha256_hash",
        "challenge": "base64_challenge",
        "response": "base64_signature"
      },
      "location": {
        "coordinates": "41.2565,-95.9345",
        "accuracy": 10,
        "address": "456 Main Street"
      },
      "carrier": {
        "driver": "driver123._dspip.usps.example",
        "signature": "carrier_attestation"
      }
    }
  ],
  "signature": "provider_signature"
}
```

6.7.3. Callback Mechanism

Senders MAY specify callback URLs for real-time confirmation:

```
"deliveryConfirmation": {
  "callbackUrl": "https://sender.com/dspip/confirm",
  "callbackAuth": "Bearer abc123xyz789",
  "callbackMethod": "POST"
}
```

Callback payload:

```
{
  "itemId": "TRACK-2025-000123",
  "status": "delivered|attempted|returned",
  "timestamp": 1703548900,
  "proof": {
    "type": "cryptographic",
    "challenge": "base64_challenge",
    "response": "base64_signature"
  },
  "signature": "provider_signature"
}
```

7. Cryptographic Operations

7.1. Key Generation

DSPIP uses the secp256k1 elliptic curve as specified in [SEC2]:

* Curve: secp256k1

* Private key: 256 bits (32 bytes)

* Public key: 264 bits (33 bytes compressed)

For split-key labels, [Ed25519] is used with 256-bit keys.

7.2. Signature Creation

Standard signature using ECDSA:

1. Construct signable content: signable = protocol | version | type | keyLocator | payload

2. Calculate SHA-256 hash of signable content

3. Sign hash using ECDSA with private key

4. Encode signature in DER format

5. Convert to hexadecimal string

7.3. Signature Verification

Standard verification reconstructs signable content, calculates SHA-256 hash, retrieves public key from DNS, and verifies ECDSA signature.

Split-key verification requires Zone B reveal and uses Ed25519 verification without DNS lookup.

8. Verification Process

8.1. Verification Algorithm

Input: QR code data string. Output: Verification result with validity status.

1. PARSE: Split QR data by pipe delimiter, validate 6 or 7 fields present
2. VALIDATE PROTOCOL: Protocol MUST equal "DSPIP", Version MUST be compatible, Type MUST equal "SHIP"
3. DECODE PAYLOAD: Base64 decode and parse JSON
4. DNS LOOKUP: Query DNS TXT record for keyLocator
5. VERIFY SIGNATURE: Verify with public key
6. PRIVACY DECRYPTION: Decrypt encryptedRecipient if last mile provider
7. REVOCATION CHECK: Query revocation list
8. OPTIONAL BLOCKCHAIN: Record custody transfer

8.2. Privacy-Preserving Delivery Protocol

The privacy-preserving delivery protocol protects recipient information during transit.

8.2.1. Protocol Overview

1. Recipient Selection: At checkout, recipient selects their preferred last mile provider (post office, corporate mailroom, residential carrier)
2. Key Retrieval: System retrieves the last mile provider’s public key from DNS or directory service
3. Encryption: Recipient’s address and delivery instructions are encrypted with the last mile provider’s public key
4. Label Creation: Shipping label shows only the last mile provider destination in cleartext

5. Transit: Package routes to last mile provider with encrypted recipient data

6. Decryption: Last mile provider decrypts recipient information using their private key

7. Final Delivery: Provider completes delivery to actual recipient

8. Confirmation: Cryptographic proof of delivery recorded

8.2.2. Last Mile Provider Registration

Organizations register as last mile providers by publishing keys:

Post office:

```
omaha-main._dspip.usps.example. IN TXT
  "v=DSPIP1; k=ec; c=secp256k1;
   p=AzmjYBMwFZfa70H75ZOgLMUT0LVVJ+wt8QUOLo/0nIXC;
   types=SHIP; auth=government;
   address=1234 Post Office Way, Omaha, NE 68102;
   coverage=68101,68102,68103,68104,68105"
```

Corporate mailroom:

```
mailroom._dspip.acmecorp.example. IN TXT
  "v=DSPIP1; k=ec; c=secp256k1;
   p=CylmZCNxGafb91I86BPhNNVV2NXYL+yu8SVQNq/2pKZE;
   types=SHIP; auth=organization;
   address=123 Business Ave"
```

8.2.3. Encryption Specification

Recipients are encrypted using ECIES over secp256k1 per [ECIES]:

1. Generate ephemeral key pair (r, R = rG)

2. Compute shared secret: S = r * LastMilePublicKey

3. Derive encryption key: K = KDF(S || R) using [SHA256]

4. Encrypt: C = AES-256-GCM(K, recipient_data) per [AES]

5. Output: R || C || tag (R: ephemeral public key 33 bytes, C: ciphertext variable length, tag: authentication tag 16 bytes)

8.2.4. Organizational Hierarchies

Large organizations MAY implement tiered decryption:

Organization Master: mailroom._dspip.acmecorp.example

* Decrypts all packages to organization
* Routes internally based on decrypted data

Department Level: engineering._dspip.acmecorp.example

* Department-specific packages only

Individual Level: john.doe._dspip.acmecorp.example

* Personal packages only

8.2.5. Directory Services

Vendors MAY use directory services to map addresses to providers:

Request:

GET https://directory.dspip.org/providers?zip=68102

Response:

```
{
  "providers": [
    {
      "name": "USPS Omaha Main",
      "keyLocator": "omaha-main._dspip.usps.example",
      "services": ["standard", "express", "certified"],
      "coverage": ["68101", "68102", "68103"],
      "publicKey": "AzmjYBMwFZfa70H75ZOgLMUT0LVVJ+wt8QUOLo/0nIXC"
    },
    {
      "name": "FedEx Omaha Hub",
      "keyLocator": "omaha._dspip.fedex.example",
      "services": ["overnight", "2day", "ground"],
      "coverage": ["68101-68199"],
      "publicKey": "BxkiXPQwFZfe80J86AOhMMVU1MWWK+xu9RVPMp/1oJYD"
    }
  ]
}
```

8.2.6. Fallback Mechanism

If decryption fails:

1. Provider attempts decryption -> Fails
2. Provider looks up sender from issuer field
3. Provider contacts sender with parcelId
4. Sender provides recipient information via secure channel
5. Delivery proceeds normally
6. Incident logged for audit

8.2.7. Customs Compliance

For international shipping:

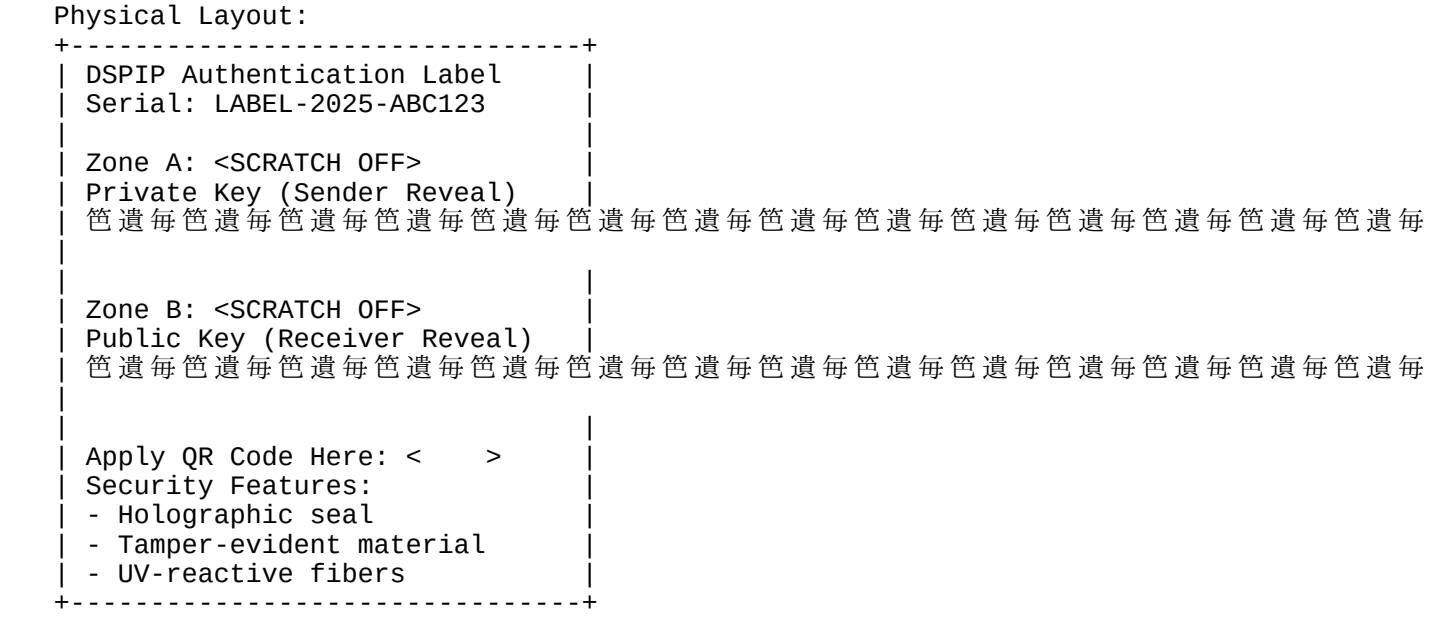
```
{
  "typeData": {
    "customsInfo": {
      "contents": "Electronics",
      "value": 500,
      "hsCode": "8471.30",
      "originCountry": "US",
      "destinationCountry": "CA"
    },
    "lastMileProvider": "toronto._dspip.canadapost.example",
    "encryptedRecipient": "base64_encrypted"
  }
}
```

8.3. Physical Authentication via Split-Key Labels

Split-key labels prevent QR code cloning through physical cryptographic binding.

8.3.1. Label Manufacturing

Secure labels with embedded key pairs:



8.3.2. Manufacturing Requirements

Label producers MUST:

* Generate unique key pairs using HSM
* Use cryptographically secure random generation
* Print using secure facilities
* Use tamper-evident scratch-off material
* Destroy all digital copies after printing
* Never retain private keys
* Maintain audit log of serial numbers

8.3.3. Split-Key Protocol

MANUFACTURING:

1. Generate keypair (sk, pk) using [Ed25519]
2. Print sk under Zone A (32 bytes hex)
3. Print pk under Zone B (32 bytes hex)

4. Destroy digital copies

5. Package in tamper-evident packaging

SENDER:

1. Purchase authenticated label

2. Verify label authenticity

3. Scratch Zone A -> reveal sk

4. Create payload with privacyMode = "split-key", authenticationProfile = "PHYSICAL-SPLIT-KEY", labelSerial = "LABEL-2025-ABC123"

5. Sign: signature = Sign_Ed25519(sk, payload)

6. Generate QR code

7. Apply to label

8. Destroy revealed sk

RECEIVER:

1. Receive package with intact Zone B

2. Scan QR -> extract payload + signature

3. Scratch Zone B -> reveal pk

4. Verify: Verify_Ed25519(pk, payload, signature)

5. If valid: Package authentic

6. If invalid: Package cloned/tampered

8.3.4. Security Properties

This mechanism provides:

* Clone Detection: Cloned QRs lack matching public keys

* Tamper Evidence: Scratch-offs show if revealed

* Non-transferable: Cannot move QR to different label

* Offline Verification: No network required

* Forward Secrecy: Private key destroyed after use

8.3.5. Anti-Cloning Analysis

Attack: Copy QR to new package

* -> No matching public key on new label

* -> Verification fails

Attack: Copy entire label

* -> Cannot manufacture same key pair

* -> Different keys = verification fails

Attack: Intercept and re-sign

* -> Private key already destroyed

* -> Cannot create valid signature

Attack: Photograph public key area

* -> Still need private key for forgery

* -> Cannot create fake packages

8.4. Delivery Confirmation Protocol

DSPIP supports cryptographic proof of delivery.

8.4.1. Challenge-Response Proof

At delivery:

1. Carrier generates: challenge = nonce || parcelId || timestamp || location
2. Carrier presents challenge to recipient
3. Recipient signs: proof = Sign(recipient_key, challenge)
4. Carrier verifies proof
5. Carrier records the confirmation

Confirmation record format:

```
{
  "parcelId": "TRACK-2025-000123",
  "challenge": "base64_challenge",
  "proof": "base64_proof",
  "timestamp": 1703548900,
  "location": "41.2565,-95.9345"
}
```

8.4.2. Single-Use Delivery Keys

Recipients MAY generate ephemeral keys:

At checkout:

```
{
  "deliveryConfirmation": {
    "publicKey": "ephemeral_pubkey",
    "validUntil": 1704153600
  }
}
```

At delivery:

```
{
  "challenge": "nonce:abc123:1703548900",
  "response": "Sign(ephemeral_key, challenge)",
  "discardKey": true
}
```

8.4.3. Proof Storage

Delivery proofs are published:

```
{
  "itemId": "TRACK-2025-000123",
  "delivered": {
    "timestamp": 1703548900,
    "recipientKeyHash": "sha256_hash",
    "challenge": "base64_challenge",
    "response": "base64_signature",
    "carrier": {
      "driver": "driver123._dspip.fedex.example",
      "signature": "carrier_attestation"
    },
    "location": {
      "coordinates": "41.2565,-95.9345",
      "accuracy": 10
    }
  }
}
```

8.4.4. Multi-Party Confirmation

For high-value shipments:

```
{
  "itemId": "TRACK-2025-000123",
  "confirmations": [
    {
      "party": "recipient",
      "signature": "recipient_signature",
      "timestamp": 1703548900
    },
    {
      "party": "carrier",
      "signature": "carrier_signature",
      "timestamp": 1703548905
    },
    {
      "party": "witness",
      "signature": "witness_signature",
      "timestamp": 1703548910
    }
  ]
}
```

9. Security Considerations

9.1. Threat Model

DSPIP addresses the following shipping-specific threats:

* Package Forgery: Mitigated through cryptographic signatures
* QR Code Cloning: Prevented through split-key labels
* Recipient Privacy Breach: Protected through encryption
* Delivery Fraud: Cryptographic proof prevents false claims
* Package Tampering: Signatures detect modifications
* Replay Attacks: Unique tracking IDs prevent reuse
* Man-in-the-Middle: DNSSEC protects key lookups
* Tracking Attacks: Privacy modes prevent correlation

9.2. Key Management

Private Key Protection:

* Shipping keys SHOULD use hardware security modules
* Last mile providers MUST secure decryption keys
* Split-key labels provide one-time-use keys

Key Rotation:

* Organizations SHOULD rotate keys annually
* Old keys remain in DNS for verification
* Emergency rotation procedures required

Key Revocation:

* Compromised keys removed from DNS immediately
* Affected packages reissued
* Revocation lists updated

9.3. Privacy Considerations

Data Minimization:

* Use encrypted mode for consumer deliveries
* Limit address fields to necessary information
* Comply with [GDPR] where applicable

Privacy Modes:

* Standard: Business shipments with transparency
* Encrypted: Consumer privacy protection
* Split-key: Maximum security for valuable items

Correlation Prevention:

* Random tracking IDs
* Limited timestamp precision
* Different encryption per package

9.4. Revocation Mechanisms

Package revocation for lost/stolen items:

Individual records:

```
_revoked._dspip.example.com IN TXT
  "v=DSPIP1; itemId=TRACK-2025-000123;
   revoked=1703548900; reason=lost"
```

Bulk lists:

```
revocation._dspip.example.com IN TXT
  "v=DSPIP1; type=revocation;
   url=https://example.com/dspip/revoked.json;
   updated=1703548800"
```

Auto-pruning:

* SHIP entries expire after 180 days

Privacy option:

* [BLOOM] filters hide specific revoked items

9.5. Operational Security

Label Manufacturing:

* Secure facilities required
* Audit trails for serial numbers
* No retention of private keys
* Tamper-evident packaging

Directory Services:

* HTTPS required
* Authentication for updates
* Rate limiting
* Cache poisoning prevention

Delivery Confirmation:

* Time-bounded challenges
* Location verification
* Multi-party attestation for high value

10. IANA Considerations

This document requests IANA to register the following:

1. Registration of "_dspip" in the "Underscored and Globally Scoped DNS Node Names" registry per [RFC8552]
2. Reservation of "DSPIP" as a protocol identifier
3. Registration of DSPIP-specific DNS TXT record tags
4. Creation of DSPIP Type Registry with initial value SHIP
5. Creation of DSPIP Privacy Mode Registry (standard, encrypted, split-key)

11. Normative References

[ECIES] Gayoso Martinez, V., Hernandez Encinas, L., and C. Sanchez Avila, "A Survey of the Elliptic Curve Integrated Encryption Scheme", Journal of Computer Science and Engineering, Volume 2, Issue 2, August 2010.

[Ed25519] Bernstein, D., "High-speed high-security signatures", Journal of Cryptographic Engineering, Volume 2, pp 77-89, DOI 10.1007/s13389-012-0027-1, September 2012, <https://doi.org/10.1007/s13389-012-0027-1>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8552] Crocker, D., "Scoped Interpretation of DNS Resource Records through "Underscored" Naming of Attribute Leaves", BCP 222, RFC 8552, DOI 10.17487/RFC8552, March 2019, <https://www.rfc-editor.org/info/rfc8552>.

[SEC2] Standards for Efficient Cryptography Group, "SEC 2: Recommended Elliptic Curve Domain Parameters", Version 2.0, January 2010, <https://www.secg.org/sec2-v2.pdf>.

12. Informative References

[AES] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001.

[Bitcoin] Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008, <https://bitcoin.org/bitcoin.pdf>.

[BLOOM] Bloom, B., "Space/Time Trade-offs in Hash Coding with Allowable Errors", Communications of the ACM, Volume 13, Issue 7, pp 422-426, July 1970.

[EDI856] X12, "Ship Notice/Manifest Transaction Set", ASC X12 Standard EDI 856, 2020.

[Ethereum] Buterin, V., "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform", 2014, <https://ethereum.org/whitepaper/>.

[GDPR] European Parliament and Council, "Regulation (EU) 2016/679 (General Data Protection Regulation)", April 2016.

[GS1-EPCIS] GS1, "EPCIS (Electronic Product Code Information Services) Standard", Version 2.0, 2022, <https://www.gs1.org/standards/epcis>.

[ISO18004] ISO/IEC, "Information technology - Automatic identification and data capture techniques - QR Code bar code symbology specification", ISO/IEC 18004:2015, 2015.

[ISO3166] ISO, "Codes for the representation of names of countries and their subdivisions", ISO 3166-1:2020, 2020.

[PLUSCODE] Google, "Open Location Code (Plus Codes)", <https://github.com/google/open-location-code>. An open-source system for encoding geographic coordinates into a compact, URL-safe string.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[SHA256] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS PUB 180-4, August 2015.

Appendix A. Test Vectors

A.1. Test Key Pair

Private Key (hex):

e8f32e723decf4051aefac8e2c93c9c5b214313817cdb01a1494b917c8436b35

Public Key Compressed (hex):

0339a36013301597daef41fbe593a02cc513d0b55527ec2df1050e2e8ff49c85c2

Public Key Base64 (for DNS):

AzmjYBMwFZfa70H75ZOgLMUT0LVVJ+wt8QUOLo/0nIXC

A.2. Standard Mode Shipping

Payload:

```
{
  "type": "SHIP",
  "issuer": {
    "organization": "ACME Logistics",
    "address": {"city": "Omaha", "state": "NE", "country": "US"}
  },
  "subject": {
    "name": "Bob Jones",
    "address": {
      "street1": "456 Main Street",
      "city": "Lincoln",
      "state": "NE",
      "postalCode": "68501",
      "country": "US"
    }
  },
  "itemId": "TRACK-2025-000123",
  "timestamp": 1703548800000,
  "typeData": {
    "privacyMode": "standard",
    "parcelId": "TRACK-2025-000123",
    "carrier": "ACME",
    "service": "Ground"
  }
}
```

Signature (DER-encoded ECDSA/secp256k1, hex):

```
30440220250f55bf60f3f82031677d17e6202fbf12e31b9ce8d1541e287e1fdd8
ce40a41022056ef5cd183a674c4f5fdc5e0cac1dcfd386de2e7d505681f83104e
2b1f53a315
```

A.3. Privacy Mode Shipping

Recipient data (to be encrypted):

```
{
  "recipientName": "Bob Jones",
  "address": {
    "street1": "456 Main Street",
    "apartment": "4B",
    "city": "Lincoln",
    "state": "NE",
    "postalCode": "68501"
  },
  "deliveryInstructions": "Use back door, code 4321"
}
```

A.4. Split-Key Mode

Label Serial: LABEL-2025-ABC123

Zone A Private Key (Ed25519, hex):

9d61b19deffd5a60ba844af492ec2cc44449c5697b326919703bac031cae7f60

Zone B Public Key (Ed25519, hex):

d75a980182b10ab7d54bfed3c964073a0ee172f3daa62325af021a68f707511a

Signature (Ed25519, hex):

1fb94d499a504e201433d0e783906a013e26c21daca3ab8c5ad1e9fcf73a1c58
c9ed3abd60802f3abaac01d35bada76ab1bf571ca1641b8d3ea62f7468fbef0f

A.5. DNS TXT Records

Warehouse:

```
warehouse._dspip.example.com. IN TXT "v=DSPIP1; k=ec; c=secp256k1;
  p=AzmjYBMwFZfa70H75ZOgLMUT0LVVJ+wt8QUOLo/0nIXC; types=SHIP"
```

A.6. Revocation List

```
{
  "version": "1.0",
  "issuer": "warehouse._dspip.example.com",
  "updated": 1703548800,
  "revoked": [
    {
      "itemId": "TRACK-2025-000123",
      "revoked": 1703548900,
      "reason": "lost"
    }
  ],
  "signature": "base64_signature"
}
```

Appendix B. Implementation Guidelines

B.1. QR Code Generation

Recommended settings:

* Error correction level M (15%) for standard labels

* Error correction level Q (25%) for split-key labels
* Error correction level H (30%) for outdoor use
* Automatic version selection
* Binary encoding mode

B.2. Performance Benchmarks

Expected operation times:

* Key generation: 5-10 ms
* Signature creation: 2-5 ms
* DNS lookup: 20-100 ms (cacheable)
* Signature verification: 3-8 ms
* ECIES encryption: 10-20 ms
* ECIES decryption: 10-20 ms

B.3. Caching Strategy

B.3.1. DNS Key Record Caching

Key records follow standard DNS caching semantics:

* Cache based on DNS TTL (recommended: 3600-86400 seconds)
* Minimum recommended TTL: 300 seconds (5 minutes)
* Maximum recommended TTL: 86400 seconds (24 hours)
* Implement negative caching for failed lookups
* Honor emergency flush requests via TTL=0

B.3.2. Revocation List Caching

Revocation lists require stricter freshness controls:

* Respect max-age field in revocation records
* Recommended refresh interval: 5-15 minutes for active verification

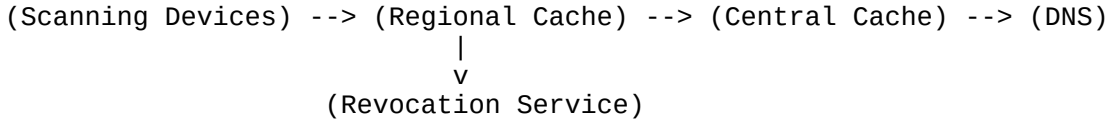
* MUST NOT serve revocation data older than max-age

* Use bloom filters for privacy when appropriate

B.3.3. Organizational Caching Architecture

Large-scale logistics operations (thousands of daily verifications) SHOULD implement centralized caching infrastructure:

Recommended architecture:



Tier 1 - Device Level:

* Hot cache for current shift’s frequent senders

* TTL: 5-15 minutes

* Capacity: 100-500 most recent key locators

Tier 2 - Regional/Facility Level:

* Serves all devices in a geographic area

* TTL: 1-4 hours

* Maintains revocation list copies

Tier 3 - Central/Organization Level:

* Single point of external DNS queries

* TTL: Full DNS TTL

* Authoritative cache for the organization

Benefits:

* Dramatic reduction in external DNS queries (95%+ reduction)

* Predictable DNS load on signing domains

* Offline resilience when network connectivity is interrupted

* Centralized revocation monitoring and push updates

B.3.4. Offline Operation

Devices MAY operate offline with cached data:

Cache Age	Verification Behavior
< TTL	Normal verification, full trust
TTL to 4 hours	Verify with CACHE_STALE warning flag
4-24 hours	Verify with OFFLINE_MODE warning flag
> 24 hours	Fail verification, queue for re-check

Table 1

Organizations SHOULD define acceptable staleness thresholds based on their security requirements and operational constraints.

Note: Revocation checks MUST use fresh data when network is available. Offline verification without revocation checks SHOULD be flagged.

B.3.5. Directory Services Caching

Directory service responses (provider lists, coverage data):

* Cache provider lists by region

* Update daily or on configuration change

* Implement failover mechanisms for service outages

Appendix C. Use Case Examples

C.1. E-commerce Order with Privacy

Customer checkout flow with privacy-preserving delivery:

1. Customer enters address, selects preferred carrier

2. System retrieves carrier’s public key, encrypts recipient

3. Label shows only carrier destination

4. Carrier decrypts at delivery, completes with proof

C.2. High-Value Package with Anti-Cloning

Split-key label flow for valuable items:

1. Sender scratches Zone A, signs package, destroys private key
2. Package transits with unverifiable signature (cloning prevented)
3. Recipient scratches Zone B, reveals public key, verifies

C.3. Corporate Mailroom Delivery

Internal routing with organizational keys:

1. Employee selects corporate mailroom as destination
2. System encrypts internal routing (dept, room number)
3. Mailroom decrypts and routes internally

C.4. International Shipping

Cross-border delivery with customs compliance:

1. Customs info public, recipient encrypted
2. Customs verifies declarations, cannot see recipient
3. Local provider decrypts and completes delivery

Author's Address

Andy Boell (editor)
Midwest Cyber LLC
1022 Brickyard Dr
Hooper, NE 68031
United States of America
Email: contact@dspip.io
URI: https://dspip.io