RSW-0003 — Identity and Signatures

Revision 0.01 | Protocol rsw/1 | Format 1 | Experimental proposal

Developed by Pablo Murad — November 23, 2026 — pablo@pablomurad.com — https://rsw.works

Experimental Project Specification. This document is an experimental specification of the Really Small Web project. It is a work in progress and has not been published as an RFC or approved as an Internet Standard.

License: specifications CC0 1.0 (public domain); future reference implementation and official tools MIT.

RSW-0003 — Identity and Signatures

Revision 0.01 | Protocol rsw/1 (proposed) | Format 1

Experimental normative proposal

Developed by Pablo Murad

Pablo Murad — November 23, 2026

pablo@pablomurad.com

01. Scope and Principles

RSW-0003 defines cryptographic identities, the representation of public keys, the signed bytes, the signature of a cartridge, and the conditions of authenticity and trust. It complements RSW-0001 (Cartridge Format) and RSW-0002 (Manifest Specification), both at revision 0.01, and the RSW v0.02 conceptual document. The protocol version is rsw/1 and the container format is 1. This is an experimental proposal, not a ratified standard nor a claim that the MVP is implemented.

MUST and MUST NOT indicate requirements; SHOULD indicates a recommendation; MAY indicates an option. Producers, validators, and readers MUST apply the previous rules before considering a cartridge authenticated. Cryptographic validation does not replace isolation, rendering security, publication policy, URI validation, or update verification.

An RSW identity requires no account, email address, domain, certificate issued by an authority, or central server. It is represented by an Ed25519 public key. Possession of the private key allows signing publications. A signature does not prove civil name, intellectual authorship, creation date, reputation, content safety, or legal authorization to distribute material.

RSW distinguishes four objects: the author's identity (public key); the publication's stable identity (key and name); the declared edition (stable identity and version); and the exact stored object (H_png). A version counter is not proof of uniqueness or novelty. There is no global identity consensus.

02. Keys and Identity Representation

The only algorithm of the rsw/1 profile is pure Ed25519 per RFC 8032, sections 5.1 and 5.1.5 to 5.1.7, with the verification restrictions of section 05 of this document. Do not use Ed25519ph, Ed25519ctx, X25519, ECDSA, Schnorr, or algorithm negotiation in a cartridge of this profile. Ed25519 uses SHA-512 internally; SHA-256 is used for the content commitments of RSW.

A new private key MUST be created from 32 random bytes obtained from the system's cryptographically secure source. Do not derive it from a password, name, email, clock, or predictable identifier. The public key A consists of 32 bytes, the compressed encoding of an edwards25519 point. The producer MUST use an established cryptographic library and fail if the random generator fails. The identifier is not a hash of the key.

The public_key field of manifest.json contains literally ed25519: followed by RFC 4648 Base32, A–Z and 2–7 alphabet, in uppercase, without padding. The Base32 part has 52 characters; the complete identifier has 60 ASCII bytes. The last character can only be A or Q (four residual bits zero). Decode exactly 32 bytes, re-encode, and compare byte by byte. Do not accept lowercase letters, aliases, spaces, line breaks, padding,

or non-zero residual bits. A correct textual encoding is not sufficient: the key must also satisfy section 05.

The author name is informative text and need not be unique. The stable identity of a signed publication is the tuple (A, name), where A is the 32 public bytes and name is the ASCII identifier of RSW-0002. The textual key identifier and name may be presented together. The external rsw://key/... grammar will be specified by RSW-0004, without presuming that the URI representation is identical to that of public_key.

The hosting domain is a location, not the identity. The same author can distribute copies on multiple domains. Two keys with the same author or name remain distinct identities. Renaming name creates another publication. An anonymous cartridge has no stable cryptographic identity and cannot replace a signed publication.

03. Canonical Manifest and Signed Message

The producer prepares the final cover per RSW-0001 and all regular files. It computes cover_hash = SHA-256 of the exact bytes of the PNG without the rsWT chunk, and files[path] = SHA-256 of the uncompressed bytes of each non-reserved file. It uses the textual values sha256: followed by 64 lowercase hex, per RSW-0002. It includes public_key before signing.

The manifest MUST pass the syntactic and semantic validation of RSW-0002, including duplicate members, limits, lexical integers, Unicode, inventory, and field relations. During production, the relation between key and signature is verified on the final package, after the signature. Absent optional fields are not inserted, strings are not normalized, and unknown extensions are not discarded. Do not sign a struct representation that has lost data.

C = JCS(manifest.json), the canonical JSON representation in UTF-8 per RFC 8785. Apply canonicalization to the complete object, not to the raw text. The integer-only numeric subset of RSW-0002 guarantees exact representation; do not accept fractions, exponents, or -0 and then convert them. Objects are ordered recursively by UTF-16 code units of the decoded member names; arrays keep order. Escape strings per JCS, without additional HTML escapes, without Unicode normalization, and without BOM. Do not append a newline, spaces, or a terminator to the result. The stored manifest may be readable or canonical, as long as it is valid. For reproducibility, the producer SHOULD store C directly.

The domain prefix P is the ASCII sequence "RSW/1:cartridge-manifest" followed by a NUL byte (0x00). It has 25 bytes. In hexadecimal:

```
5253572f313a636172747269646 7652d6d616e696665737400
```

The message M is P || C, a byte concatenation without additional length or trailing separator. The domain is exclusive to the signature of rsw/1 cartridge manifests. Indexes, events, challenges, delegations, or other future messages MUST use distinct domains defined in the corresponding specifications. The prefix is never extracted from a cartridge-controlled field.

The signature is computed directly as Ed25519.Sign(seed, M), using the appropriate private-key API of the library. Do not sign SHA-256(M), SHA-512(M), the ZIP hash, a hexadecimal string, or Base64(M). The SHA-256 hashes of M and of C MAY be reported for diagnostics and vectors; they do not replace M in the algorithm. A producer need not implement Ed25519 internally.

The commitment authenticates the cover and the inventoried files, plus all present manifest fields, including version, previous_release, public_key, peers, and extensions. Changing any committed value invalidates the signature, except JSON representation changes that preserve C. No network operation or execution is authorized by a signed field.

04. Representation of signature.sig and Integrity

signature.sig is a regular file at the root of the ZIP, exactly 64 binary bytes: R (32 bytes, compressed point) || S (32 bytes, little-endian integer). Without header, algorithm, repeated key, ASN.1, hexadecimal text, Base64, line breaks, BOM, or additional data. The ZIP method can be Store or Deflate; the length

comparison is done after expansion. This special limit refines the structural 1..1024 byte limit of RSW-0001.

The public_key key and signature.sig MUST be both present or both absent. The presence of only one makes the manifest invalid. A cartridge with both absent can be imported locally as UNSIGNED, as long as it satisfies all other rules. A malformed, wrong-length, or invalid signature MUST NOT be reinterpreted as absence of a signature.

manifest.json and signature.sig are reserved and excluded from the files inventory. The whole manifest is authenticated by M; the signature is verified cryptographically. Do not include the hash of either file in files. H_png and H_zip do not appear in the manifest as hashes of themselves. previous_release, when required, identifies the previous edition, not the current one.

H_cover binds the original bytes of the cover, including headers and CRCs; comparing appearance or pixels is not sufficient. H_png is the SHA-256 of the complete PNG and identifies an exact copy. H_zip is the SHA-256 of the raw ZIP payload. These are distinct hashes. Because C and the uncompressed files are signed, a change of order, compression method, or JSON representation that keeps the commitments may preserve the signature but produce another H_png. The new object is not the same exact copy. RSW-0013 will decide how to treat different packagings of the same edition and content conflicts. Until then, do not silently select one as an update.

05. Verification Profile and Algorithm

To prevent degenerate keys and divergences between Ed25519 libraries, the profile requires strict point encodings and the cofactorless equation. Let p = 2^255 - 19 and L = 2^252 + 27742317777372353535851937790883648493. B is the RFC 8032 base point and O is the group identity.

A and R MUST be decoded per RFC 8032, section 5.1.3: y coordinate less than p, a valid point on the curve, sign bit zero when x is zero, canonical 32-byte encoding. A MUST be different from O and belong to the prime-order subgroup: [L]A = O. R MUST belong to the same subgroup; R = O is allowed. S must satisfy 0 <= S < L. The subgroup test must perform a multiplication by the integer L without reducing it modulo L first; reducing the scalar would render the test useless. Verify the equation [S]B = R + [k]A, where k is the little-endian integer from SHA-512(R || A || M), reduced modulo L. Do not accept a permissive verification that ignores these conditions.

The definition is normative, not a recommendation to build one's own cryptographic primitives. Use audited APIs for point decoding and validation, in addition to the Ed25519 verifier. The zero-dependency goal is secondary to correctness; a small, auditable dependency may be necessary. With prime-order A and the cofactorless equation verified, the subgroup condition for R follows from the equation; an implementation may use this equivalence. Do not trust a library result without knowing its acceptance policy. Do not use variable-time comparisons for secrets; the points verified here are public.

Reading algorithm:

1. Validate the PNG/ZIP structure per RSW-0001, with resource limits and CRCs. Preserve the original bytes.
2. Validate the complete manifest per RSW-0002, including inventory, real file hashes, H_cover, and consistent presence of key and signature. Do not render partial content.
3. If both reserved authentication fields are absent, return UNSIGNED. Otherwise, decode the canonical public key and verify the conditions on A.
4. Read the 64 bytes of signature.sig, verify R and S, build C and M with the fixed prefix.
5. Verify Ed25519 per the strict profile. If it fails, return INVALID, with a diagnostic code. Never try another canonicalization, algorithm, prefix, key, or version to obtain success.
6. When the operation has an expected reference, verify that the key, name, and/or H_png match the expectation exactly. A valid signature from another identity does not satisfy the requested reference. Index

authentication and reference resolution belong to RSW-0005/0006.

7. If all applicable conditions are satisfied, return VERIFIED or TRUSTED according to local policy. Apply the version, rendering, and network rules separately. Signature success does not authorize opening scripts or external access.

Additional errors: INVALID_PUBLIC_KEY (invalid encoding or point); INVALID_SIGNATURE_ENCODING (invalid length, R, or S); SIGNATURE_MISMATCH (equation not satisfied); IDENTITY_MISMATCH (expected key or name divergent); OBJECT_HASH_MISMATCH (expected H_png divergent). INVALID_MANIFEST and INTEGRITY_MISMATCH remain per RSW-0002. A reader without cryptographic support must indicate SIGNATURE_PENDING or absent support, never VERIFIED. Identical diagnostic order is not required when there are multiple failures; acceptance must coincide.

06. Trust, Publication, and Authentication Limits

Reader states:

```
UNSIGNED — valid structural integrity and manifest, without signature.
SIGNATURE_PENDING — cryptographic verification not yet executed.
VERIFIED — valid signature and commitments; key not locally trusted.
TRUSTED — same requirements as VERIFIED, plus explicit local trust.
INVALID — format, integrity, or signature failure; do not render.
```

Unknown profiles or versions and policy refusals are separate states, not equivalent to an invalid signature. The interface must distinguish cryptographic authenticity, unconfirmed human identity, local trust, and execution safety. Trust is not transitive, is not transferred by peers, and implies no additional permissions.

The reader may register a trusted key by explicit action and show its complete identifier for verification through an independent channel. Do not automatically trust an author, title, domain, cover image, or key presented in another cartridge. Trust must be bound to the key bytes, not to the spelling of a name. Removing trust is a local decision and does not invalidate previous signatures.

Unsigned imports are allowed in the MVP only as local files with a warning; they must not be accepted as authenticated network publications nor update signed identities. A publication by domain requires an appropriate binding among index, key, and name; TLS authenticates the domain, not the person who signs the content. The first domain/key association is not proof of human identity; stay alert to unexpected key changes. The exact rules are in RSW-0005/0006.

A signature does not prove that a version is recent. created_at is an author declaration, not a reliable timestamp. The reader must separate authentication from currency, keep the largest known authenticated edition, and detect regressions per future RSW-0013. Without external information, it cannot know whether the key was compromised or whether a newer publication exists. RSW-0015 will define rotation, recovery, and revocation; there is no universal revocation of existing copies.

07. Generation, Custody, and Publication Flow

The MVP does not standardize an interoperable private-key container; the reference software may keep the seed in a local versioned format, protected by the operating system. No seed, expanded private key, protection password, or recovery secret may enter the cartridge, index, log, production fixture, or web interface. The public key is distributable. Using a credential manager or encrypted storage is recommended when available.

On Unix, restrict the key directory to the owner (0700) and files to the owner (0600), verify the result, do not follow symlinks when opening/creating, and avoid broad permissions during creation. On Windows, apply the equivalent ACL; do not assume chmod is sufficient. Protect backups and instruct the user to keep them offline. Do not send the private key to hosting servers. No endpoint accessible to cartridge content may export or use private keys. In case of loss, a new key creates another identity; there is no recovery by email. In case of suspected compromise, stop use and communicate through independent channels.

Production order: prepare the final cover and resources; validate limits and paths; obtain the public key; build the manifest with hashes; canonicalize; sign; write signature.sig; assemble the ZIP in the RSW-0001 profile; insert rsWT and CRC; validate the complete package; compute H_png; publish the original bytes and references per the future hosting specification. A producer must prevent publishing a partially valid cartridge.

Conceptual commands of the Go executable, not yet implemented:

```
rsw keygen
rsw identity
rsw pack ./hello --cover cover.png --out hello.rsw.png
rsw verify hello.rsw.png
rsw inspect hello.rsw.png
```

The default publication mode must require a user-chosen key. Do not automatically sign received content without explicit action. An eventual private-key export command must be local, explicit, and protected. In Go, crypto/rand, crypto/ed25519, crypto/sha256, and encoding/base32 provide the foundation; a JCS canonicalizer per RFC 8785 and strict point/JSON validation complete the necessary checks.

08. Vectors and Interoperability

The vector set delivered with this revision uses a public test key and a fixed seed exclusively for reproduction. The project's textual credit does not mean that this key belongs to Pablo Murad. It MUST NOT be used for real publication, production trust, or identity recovery. The example cartridge does not replace the RSW-0001 and RSW-0002 structural vectors.

The vectors include: signed cartridge; reformatted manifest with the same signature; altered manifest without updating the signature; altered resource with correct ZIP CRC; altered signature; truncated signature; valid inventory with degenerate public key; non-canonical Base32 encoding; unsigned cartridge; S out of range; signature with the wrong domain. The bytes C and M, key and signature in hexadecimal for diagnostics, and expected results are also provided. Diagnostic formats are not the signature.sig format.

For full conformance, two independent implementations must obtain the same bytes C and M, accept and reject the same vector set, and compute the same hashes. Tests must be added for points off the curve, non-canonical points, torsion and mixed order, S >= L, signature from a different domain, wrong keys, UTF-16 out of code-point order, equivalent Unicode and escapes, duplicates, and forbidden numbers. Library cryptographic tests do not certify the PNG/ZIP parser, reader security, or the full strict-profile implementation.

09. Remaining Work and References

This specification fixes the cartridge identity and signature scheme, but is still experimental. Pending: implement strict parser, complete verifier, and key management in the Go core; select auditable dependencies; publish edge-case and fuzzing tests; measure resources; validate full interoperability; ratify the rules with RSW-0004 (URI), RSW-0005/0006 (resolution, signed indexes, and hosting), RSW-0011 (security), RSW-0013 (versions), and RSW-0016 (compatibility).

Outside the MVP: key rotation/delegation, distributed revocation, offline-key recovery, Web of Trust, federated identities, PGP/Nostr integration, event signatures, message encryption, remote signing, mandatory HSM, and additional algorithm formats. Ed25519 is not a message-encryption key, nor is it Nostr's Schnorr scheme. Do not automatically convert or reuse keys from other protocols.

Internal references: RSW v0.02, RSW-0001 and RSW-0002, revision 0.01. Technical references:

```
RFC 8032 — https://www.rfc-editor.org/rfc/rfc8032.html
RFC 8785 — https://www.rfc-editor.org/rfc/rfc8785.html
RFC 4648 — https://www.rfc-editor.org/rfc/rfc4648.html
RFC 8259 — https://www.rfc-editor.org/rfc/rfc8259.html
Go crypto/ed25519 — https://pkg.go.dev/crypto/ed25519
FIPS 180-4 — https://csrc.nist.gov/pubs/fips/180-4/upd1/final
```

License: CC0 1.0 (public domain dedication; see CC0-1.0.txt in the downloads area). The document remains an experimental specification; the license is ratified. Next specification: RSW-0004 — URI Scheme.

Developed by Pablo Murad | pablo@pablomurad.com

A. Informative Reference Vector

File: RSW-0003-exemplo.rsw.png. Size: 1,034 bytes. The vectors ZIP package contains 11 cartridges, three positive and eight negative, reference bytes, expected results, and two independent component tests. The test key has a public seed; never use it in production.

```
Test public key:
ed25519:AOQQPP7TZYIL4HLQ3UMOOS6ATFT6JVRQTOSQ2XY53SDGIESVGG4A

SHA-256 of the canonical manifest:
sha256:1cb4f737d008f9f837444df60aee0f23cde26c1a34e5b9c8f03eca555e9010f0

SHA-256 of the signed message:
sha256:475a90a477a8659c00c9e45d8e2e0b1ca17c7391eafdedc536a9494c9013a330

Signature in hex (diagnostics only, 64 bytes):
3b33f31f54b035c269b4c40896d2e77e77d422bf35236894d562b51897abc93e
7559b7fd30a55512a7ca05150cb2befd80a625d3f1627c74a2857a47e6bcb708

Hash of the complete PNG:
sha256:a5ff1a294b1c1a1b7dadc55acca226d164acbf74e4d5da9552635f17a7b47d26

Hash of the ZIP:
sha256:b4852bb2e49e23f976ab63d7e4c57be07ec9db1487216def2c357120c1235888

Hash of the cover:
sha256:e878950f8091ec010cf5cc723bdea027a8539cf7147cfea199c2f666232dcd4e
```

Verifications performed: opening the 11 PNG covers, chunk CRCs, and ZIP tests; generation and verification of the positive signature with Python/cryptography; independent canonicalization of the manifest with Node.js (including UTF-16 ordering of supplementary characters); independent verification of the positive signature and rejection of an altered message with Go crypto/ed25519. The remaining results of the set are normative expectations for the future complete validator; they do not represent full execution of the strict profile.

Developed by Pablo Murad | pablo@pablomurad.com END — RSW-0003

Author's Address

Pablo Murad
Independent
pablo@pablomurad.com
https://pablomurad.com
Official project website: https://rsw.works

Small files. Independent worlds.